Biologically Inspired Algorithms for Wireless Sensor Networks

by Roberto Pagliari

BIOLOGICALLY INSPIRED ALGORITHMS FOR WIRELESS SENSOR NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Roberto Pagliari

August 2010

BIOLOGICALLY INSPIRED ALGORITHMS FOR WIRELESS SENSOR NETWORKS

Roberto Pagliari, Ph.D.

Cornell University 2010

In recent years, several models introduced in mathematical biology and natural science have been used as the foundation of networking primitives. These bio-inspired algorithms often solve complex problems by means of simple and iterative local rules. In this work, we consider the design and development of novel decentralized algorithms for distributed systems, with applications to wireless sensor networks, wireless body area networks and formation control.

We consider two models of interaction. In one model, nodes communicate via pulses whose arrival time is sensed and compared to a local state variable and triggers an appropriate local update. In the second model, the nodes can exchange integer valued messages, which we call *colors*.

The first class of algorithms falls in the class of Pulse-Coupled Oscillator (PCO) models that were first introduced in mathematical biology and that have been recently introduced in the sensor networking area. This thesis is concerned with the design and analysis of PCO based protocols for synchronization and multiple access.

The second class of protocols relates to the so called *voting models* introduced in Physics. The protocol was proposed for network control in particularly harsh media, where communications are severely limited by the significant distortion and delay of the link.

BIOGRAPHICAL SKETCH

Roberto Pagliari was born on February 3rd, 1980. He received his "Laurea" Degree in Telecommunications Engineering in 2005 from the University of Parma, Italy. After spending a brief period at the University of Parma and one year at the University of Genova as a research associate, he joined Cornell University in 2007 as a graduate student, where he obtained his M.Sc. and Ph.D. in 2009 and 2010, respectively. His research interests include distributed protocols for sensor networks and gossiping algorithms.

This document is dedicated to all my Friends.

ACKNOWLEDGEMENTS

It has been a pleasure to work in a number of exciting and intellectually stimulating environments, with a wonderful array of mentors and colleagues. I would like to thank Professor Gianluigi Ferrari for his support during my Master Thesis at the University of Parma. Back in the past, he took me for the first time into the world of research, and encouraged me to further pursue my studies in a graduate program.

I thank my Advisor, Professor Anna Scaglione, for giving me the opportunity of joining the graduate program at Cornell and for her support throughout my studies. Over the years she has imparted to me part of her knowledge. Her way of thinking and her guidance have had a significant impact on my approach to scientific research.

I am immensely grateful to all people who supported me during these years, including my former college mates in Parma, my friends in the US, my parents and my brother. It is because of them that my years spent on this work have been such a delight. Finally, I would like to thank Cornell University, NSF and Welch Allyn for supporting financially my graduate studies.

Part of this thesis represents joint work with my Advisor, Professor Anna Scaglione, Yao-Win Peter Hong (NTHU), Kristi A. Morgansen (UW), Tara Javidi (UCSD), Asuman Ozdaglar and Mehmet E. Yildiz (MIT), Hamid Krim (NCSU) and Shrut Kirti (Cornell).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Long before man-made communication network existed, natural phenomena had ways of creating what we can call *order*. The spontaneous tendency towards self-organization has been observed by many scientists in the past. To name a few, spontaneous magnetization, crystallization, percolation in random media, the Belousov-Zhabotinsky reaction and the formation of structures in the universe are the most evident macroscopic manifestations of such tendency in the domain of physics. In biology, several systems showcase self-organization to some extent, including (but not limited to) protein folding in the DNA structure, flocking of birds and fishes, coordination of the human movements, social interactions amongst animals belonging to the same specie. It is reasonable to think that all the aforementioned phenomena are "fairly good" from an engineering perspective, since they are the result of millions of years of evolution.

1.1 Thesis Statement

In the past, many computing primitives were borrowed from Nature to solve complex problems. To name a few, neural networks have been extensively used in the past for pattern recognition [11]; Genetic Algorithms [25] for search, Particle Swarm Optimization (PSO) [21] and Ant Colony Optimization (ACO) [20] for optimization and networking problems.

In this thesis we study two main classes of algorithms inspired by biological systems, namely, the Pulse-Coupled Oscillators (PCO) and Nonlinear Voting models. We propose novel algorithms inspired by such mechanisms with applications

to synchronization, scheduling and formation control in wireless sensor networks. We show their convergence properties from a theoretical point of view and verify them both numerically and experimentally. We finally show that the protocols we propose have advantages over the traditional schemes proposed in literature that mainly lie in the speed of convergence, adaptability, and robustness to noisy communications.

1.2 Contribution

The main results we obtained are summarized in the following.

- **Synchronization:** we derived new convergence results about the PCO algorithms, one based on a probabilistic argument, and another based on a geometric one. A PCO software library for CC2420 transceiver, compatible with the 802.15.4 standard, was developed under the TinyOS environment; our results show the efficacy (and better scalability) of our protocol with respect to the one originally proposed in [85].
- **Scheduling in small networks:** we extended the work done in [56, 68], showing that the PCO with negative coupling can only achieve weak convergence to round-robin scheduling. We also proposed novel algorithm that provides strict convergence and proportional fairness. We then discussed a possible implementation of such primitive by integrating both the physical and mac layers into a cross-layered fashion, based on the hardware architecture developed in [5].
- **Formation control:** a radically different consensus protocol has been proposed. Such algorithms is different in the way information is processed, since

the state of a node in the network is chosen according to a probabilistic rule whose outcome depends on the state of the neighborhood. The proposed protocol is robust to communication channel non-idealities, mobility and, differently from other schemes proposed in literature [3, 12, 87], it does not require information about the degree of the neighbors nor the topology of the network.

1.3 Related Work

In the 17th century, the dutch physicist Christian Huygens studied the synchronization of two penduli mounted on the same beam, which he attributed to the "imperceptible motion of the air" [10]. Centuries later, in 1975, Charles Peskin [70] modeled the sinoatrial node cells that are responsible for the pumping of the heart as leaky integrators (a common resistor-capacitor series) followed by a threshold element; as the voltage across the capacitor reached a threshold, each cell would broadcast to its neighbors a current, perturbing the voltage at the other capacitors. The Pulse Coupled Oscillator (PCO) model was born. Since then, the model has become extremely popular in basic science and has offered insights on a number of phenomena, as is beautifully narrated in the book by Steven Strogatz [56]. Less obvious to the engineering arena was the idea that the PCO model offered a remarkably simple architecture suitable for self-organization in small networks of simple low-cost devices, as demonstrated in the works on network synchronization [29, 50, 82, 85] and distributed estimation [9]. It is not only the physical simplicity of the mechanisms that are striking, but also how interactive PCOs can achieve communication and computation in an intertwined and inseparable fashion. While there are compelling engineering reasons to employ the layered

architecture in most modern communication systems, such as the ease of interoperability among different functionalities and the flexibility in system design, these examples from mother nature show that specific applications can be achieved in a simple and efficient manner by combining different functionalities and treating them jointly as a single module.

Over the last decade other bio-inspired algorithms have been proposed in the literature as possible solutions for complex problems, most notably, the average consensus protocol, the Kuramoto model, Particle Swarm Optimization (PSO) and Ant Colony Optimization. The linear average consensus protocol, based on linear local update of the state variable, has been recently proposed in the literature, both in synchronous [13, 24, 35, 64, 87] and asynchronous [12, 23, 87] domains, as a simple primitive for global agreement, for a number of different applications ranging from synchronization, to localization and estimation. From an algorithmic perspective, the main limitation of such protocol is that the nodes are required to know their neighbors degree, that limits their use to static networks. Furthermore, the coupling is directly related to the maximum degree of the network, or to the maximum degree of the neighborhood, which is equivalent to saying that it requires global knowledge. In mobile scenarios, nodes degrees change over time, and it becomes even more difficult to guarantee a certain level of quality of service, especially considering the limited resources in terms of rate of information update, precision of state updates, and communication energy [42]. The Kuramoto model, a nonlinear version of average consensus, instead, guarantees convergence to consensus only in the case of complete networks in time, which is its major limitations since this is often not the case in mobile networks. Other protocols such as PSO and ACO have been used for network optimization and routing. However, their main limitations lie in the fact that the choices of the many parameters involved in such algorithms

are provided heuristically and it is not always clear how to set those quantities based on the specific scenario and/or application.

1.4 Outline

This thesis is organized as follows: Chapter 2 explains how the PCO works, its convergence properties and present the results we obtained by numerical simulations and experimentally. In Chapter 3 we described algorithms for the decentralized establishment of a time-division multiple access regime in small networks. These schemes have been inspired by the classic PCO model and provide a robust solution for the proportional fair allocation of a shared resource. We further describe the algorithmic aspects of such primitives and discuss their implementation at the physical layer. In Chapter 4 we introduce a novel algorithm for consensus in distributed systems, based on a randomized rule of the local node state. We derive convergence results and discuss its application in the context of formation control. Chapter 5 conclude the thesis.

CHAPTER 2

SYNCHRONIZATION BASED ON PULSE-COUPLED OSCILLATORS

2.1 Motivation and Related Work

Sensor networks synchronization algorithms belong to two main classes. The first class of methods is master-slave; it is well represented by the Network Time Protocol (NTP), that requires the flooding of a message from a master node with accurate time information over the entire network [54, 80]. The message contains in its payload a time-stamp of the transmission time of the master, measured with its absolute time reference clock, typically coming from a GPS receiver. After receiving this message every node set its internal clock to that value, correcting for delays due to multi-hop transmissions and medium access waiting times. The second class of algorithms is decentralized. A good representative is the Reference Broadcast Synchronization (RBS) [22] protocol, which is initialized by the transmission of a reference signal to the neighborhood; subsequently, the nodes exchange a time-stamp of their reference signal reception time and then compute their relative clock difference (improvements of RBS are in [78] and [57,73]). The decentralized computation of the clock skew can be performed by calculating the clock average via an iterative average consensus protocol, as suggested by [46]. We consider, instead, a radically different communication scheme to achieve synchronization in a decentralized fashion, inspired by the Pulse Coupled Oscillator (PCO). First introduced in 70s and 80s [16,28,71], the non-linear dynamics of large populations of coupled oscillators (PCO) were studied to describe the synchronous fireflies flashing, observed in the south east of Asia since the past two centuries.

A protocol imitating the PCO model was first proposed in [30] for the synchronization of wireless networks (the journal version of the article is [29]). Since [29], other authors have worked on the same problem from the theoretical point of view [34, 50, 83] and on the implementation of PCO on sensor platforms [53, 85]. The original PCO algorithm [29] makes a much more liberal use of the physical communication constraints that acknowledged possible in traditional packet-switched point to point network models. Recent work in the circuit area [84] also is providing Utra-Wideband PCO radios that emulate the original proposal in [29]. Due to the objective difficulty of interpreting the method through the usual *networking* point of view, the PCO has so far been an outlier in the context of network synchronization protocols. From the theoretical point of view, [50] mapped the algorithm into average consensus network dynamics. The mapping is in continuous time (it is not event-based) and the methodology used in [50] borrowed from [31] requires a number of approximations to hold that are neither realistic nor necessary to observe the PCO convergence. Recent results in [52] show that the approximate model used in [50] to prove convergence does not, in fact, warrant convergence for all connected networks. From the protocol implementation point of view, [85] was the first work that, rather than building a custom radio, adapted the PCO scheme to work on a common CSMA packet switched network of MicaZ Motes, as an application layer function. The effort resulted in an implementation that, as the authors recognize, is not competitive compared to the preexisting state of the art on network synchronization. More specifically, using a period set to $T = 1$s, and properly chosen values of the coupling strength ε, the PCO-based protocol in [85] was found to reach synchronization within an interval of a few milliseconds, over a network of $N = 25$ nodes, after a number of iterations ranging from 284.3 to 1164.4. These numbers do not make a software implementation of PCO a serious

contender for network synchronization.

Unlike the work in [85], our new PCO implementation is a fully integrated synchronization and MAC software primitive. The PCO, among other things, disables the Carrier Sensing MAC (CSMA) access scheme from governing the transmission times, allowing the nodes transmissions to coalesce. Our experimental results suggest that this change is critical to the success of the PCO strategy: in fact, as the nodes aggregate into groups that fire at the same time, the protocol absorbs them into a single node with "increased" power speeding up the convergence to synchrony. While PCO, like any other protocol, is adversely affected by low connectivity, it ensures the impossibility of congestion problems in the synchronization by construction, since there is no back-off mechanism in place while active. This is possible because, under our protocol implementation, concurrent transmissions are correctly handled by the standard radio interface as a single group signal that provides an input update. Nodes are therefore allowed to transmit concurrently with no delay, since there is no collision in transmitting simultaneously. This feature, that better reflects the original model [16,28,71], is lost in the implementation proposed by [85]. Thanks to our mapping and to our implementation, the comparison with other protocols comes at ease and provides very favorable results. In particular, in addition to comparing our protocol with [85], we compared the PCO to the RBS protocol combined with an average time-stamps consensus phase, obtaining the expected advantages for PCO. Unlike other protocols, PCO needs to alternate its use of the medium with other MAC mechanisms that allow peer-to-peer communications.

2.2 PCO based Synchronization

In the PCO scheme, each node has a clock, whose counter raises from 0 to T with constant speed. To simplify the exposition, it is introduced the so called *phase* variable, defined as

$$\Phi_i(t) = \frac{t}{T} + \phi_i(0) \bmod 1 \tag{2.1}$$

which reads as the local time of node i normalized by the period T modulo 1, and $\phi_i(0)$ is an initial time offset. Equation (2.1) is sufficient to describe the state of node i at any time $t > 0$, since the clock of a node can be represented as a dot on a circle of unitary length moving clockwise at constant speed. Whenever a node, say node i, crosses the finish line, i.e., $\Phi_i(t) = 1$, the node emits a message (ideally, a spike) and resets its phase to zero an instant later. There is no information encoded within the message, but, rather, the information about its state is embedded in its firing time. Any other node in the network that hears the message from node i updates its local clock or, equivalently, its phase according the following updating function:

$$\Phi_j(t^+) = \min\{f^{-1}(f(\Phi_j(t)) + \varepsilon), 1\} \tag{2.2}$$

where $\varepsilon > 0$ is called *coupling strength* and $f(\cdot)$ is a concave-down function such that $f(0) = 0$, $f(1) = 1$ and $f(x) > 0, \forall x \in (0, 1)$. Equation (2.2) seems a mathematical artifact, if we do not consider where the main model comes from. In mathematical biology [71], this model has been used to explain the synchronous pattern of pacemaker cells that, supposedly, follow a similar model to produce the *tempo* that keeps us alive. Each cell is, in fact, modeled as an oscillator, with a local phase variable $\Phi_j(t)$, as defined above. The potential of its membrane, called x_j for simplicity, depends on the phase through a nonlinear function $x_j = f(\Phi_j)$, as shown in Figure 2.1. If node j receives a message at time t from node i, node

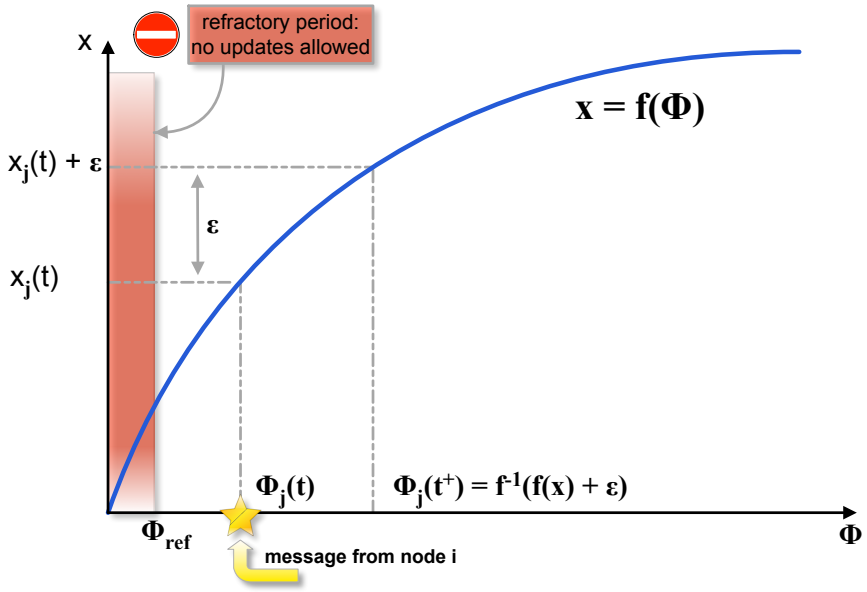

Figure 2.1: General model of PCO updating dynamics, borrowed from mathematical biology.

j updates its local phase according to (2.2), which corresponds to increasing the potential by an amount ε. Consequently, the next firing time of node j is set $(1 - \Phi_j(t^+))T$ seconds apart from the reception time, since $(1 - \Phi_j(t^+))$ is the normalized distance to the next firing time.

The nonlinear function relating the state x_j and the corresponding phase Φ_j has to be concave-down. A typical example of such function is the integrate-and-fire model, proposed by Peskin [71], or the Strogatz model [56], given by

$$x_j = f(\Phi_j) = \frac{1}{b}\log(1 + [e^b - 1]\Phi_j) \tag{2.3}$$

where $b > 0$ is a parameter that controls the curvature of $f(\cdot)$. With this particular choice of $f(\cdot)$, the update of a generic node j hearing a message at time t becomes

$$\Phi_j(t^+) = \min\{a_1\Phi_j(t) + a_0, 1\} \tag{2.4}$$

where $a_1 = e^{b\varepsilon}$ and $a_0 = \frac{e^{b\varepsilon}-1}{e^b-1}$. If the update of node j is such that $\Phi_j(t^+) = \min\{f^{-1}(f(\Phi_j(t)) + \varepsilon), 1\} = 1$, node j is said to be synchronized by node i and, thus, their clocks will match. Clearly, because of propagation delays, two nodes can never be exactly absorbed, as the time of message transmission is not equal to the time at which is received and processed. To force a stable behavior one can extend the interval in which the receiver is off for each node beyond the strict transmission time. This extra interval is called the *refractory period*, and nodes absorbed are not transmitting at unison exactly, but within a refractory period from each other [29].

2.3 Main Features of the PCO Protocol

The PCO scheme we propose presents some important features that make it different, in principle, from other communication schemes in packet switched networks.

The clock information t_i is signaled through the time of transmission. Different firing times mean that the clocks of the corresponding nodes are not synchronized. As the nodes are driven to pulse in synchrony, the method aligns the local counters progressively for the entire network and therefore provides a global clock, modulo the PCO period, that the nodes can use to perform other activities in concert. Hence, *no further information needs to be exchanged to establish a clock* if synchronization modulo the PCO period is deemed sufficient. Furthermore, the process is gradual and if the synchronous state is perturbed, the system can resume its synchrony faster then it would when it is initialized with a completely independent set of clocks.

Any group of synchronized nodes transmits PCO packets at unison. As the

time evolves, nodes are *absorbed*, i.e., they follow the same dynamics and they fire together. Ideally, groups of synchronized nodes would act as a single node, whose power is the sum of the powers of all nodes in the group. Although, in practice, the superposition of identical signals will not be additive in some cases, they usually do not interfere destructively, and the receivers are often able to decode the message sent by multiple sources at the same time, or simply detect its presence, for the following reasons.

1) **Half-duplex contraint:** every packet received within the refractory period has no impact - that is equivalent to extending "half duplex" constraint, so if or not the packets are decodable then it is irrelevant. Hence, as the packets cluster in the refractory period the protocol is not really bound to decode these packets because they fall in a period where they should not be heard in that interval anyways.

2) Outside the refractory period the packet can still be decoded for the following reasons:

- **Long PCO period:** things are initially spread out in time, as the period is much longer than the length of the packet, so the packet simply do not overlap in time.
- **Opportunistic Cooperation:** if nodes absorb each other, they may align sufficiently (a few symbols apart) to be decodable since they are superimposed identical signals which produce the same effect as multi-path [38,75,76].
- There are **capture effects**, so one packet may have significantly higher power than the other and therefore still be decoded.

The other important aspect is that, just like in RBS, the algorithm does not need every packet ever sent to be decoded, but just a sufficient number to make

progress. Because, unlike RBS, as the nodes align PCO does not need to actually decode those packets, this makes PCO more immune to congestion. This was our prediction and our objective was to illustrate that by implementing the PCO they way we did. Our convergence results are physical proof of that.

Since the PCO protocol aggregate signals and **does not suffer from backlog** as we just explained, the protocol scales favorably and performs more reliably and faster in denser networks. However, PCO operations and other transmissions need to be properly separated, because a contention resolution mechanism that is inessential for PCO messages, is still necessary to multiplex PCO signals with other information messages that have to be exchanged in the network operations.

As mentioned in the last point, the PCO primitive needs to be **multiplexed** with the regular network operations. The fact that the time of arrival is used to estimate the firing time and the absence of a backoff mechanism make the PCO signals special: not only they need to have priority over others, but any other message sent while PCO is sensing the channel is effectively creating something that can be compared to noise. While the deterioration of PCO is going to be mild for very low traffic and small coupling, it is certainly better to manage the problem.

2.4 Management of the PCO primitive

The PCO protocol offers a primitive to reach pulse synchrony, or, equivalently, an agreement on the firing of the clock. As explained in the previous point, the aligned local counters are the global clock and *no further information exchange* is needed to establish that. This primitive is useful especially in sensor networks

to coordinate the monitoring activities, and for planning contention-less interactions among the nodes. Mixing the proposed PCO with other asynchronous traffic without any management, would slow down operations and reduce the cooperative gains harvested when the nodes are absorbed. The additional network traffic would experience large update delays and strong interference, which in turn might prevent the PCO protocol from working efficiently. To prevent these undesired effects, a preferable option is to use the PCO timing to duty cycle between the activities of keeping the network synchronized and doing everything else. The idea is to prevent nodes from scheduling the transmission of information packets during the time required for the PCO firing message, the refractory period and an extra portion of the period that is needed to maintain the state of synchrony and process the PCO message. In fact, even if the network is in a state of perfect synchrony, eventually the clock drift will move the local clocks ahead or behind relative to each other. Therefore, a modest adjustment will always need to be performed. However, once synchrony is attained, the bulk of PCO signals is not going to drift away considerably, making it possible to alternate a short fraction of the period dedicated to PCO only messages (of which the nodes are globally aware by virtue of their PCO clock) followed by the regular network activities. We propose to distinguish two modes of operation: bootstrap phase and maintenance phase. Firings in both phases should have a specific indicator in the type of firing message, "PCO bootstrap" and "PCO maintenance". At deployment, the network needs to start with a sufficiently long PCO phase (as we observed in our experiments this can last a few seconds) to bootstrap the system. This is the phase marked by having a "PCO bootstrap" message included in the payload of the PCO firings. In this phase the nodes do not transmit regular traffic but only perform the PCO for a fixed number of cycles. After their completion the nodes move

into maintenance phase. The transition from the bootstrap to the maintenance phase should be gradual and marked by the completion of a the prescribed number of PCO bootstrap messages followed by a decentralized test of synchrony, where each node checks for the absence of PCO messages in the designated window for normal network traffic for a predefined number of periods. In the steady state of the maintenance phase, a large fraction of the PCO period should be dedicated to normal traffic. This normal traffic window ought to be increased progressively, as the bootstrap phase reaches completion, and the synchrony tests continue to succeed over a wider fraction of the PCO period. The percentage of the PCO period dedicated exclusively to PCO traffic should be a function of the maximum expected clock drift during the PCO period. In essence, in the PCO maintenance phase nodes are expected to be progressively drift out of sync because of the local counters drift, that determines a difference in the pace at which they which can bounded beforehand knowing the hardware clock specifications and the PCO period. Nodes can be added later to the network, or may need to restart operations, while all other nodes have completed their bootstrap phase. If a node in bootstrap phase receives a message that it is not consistent with its own state, it should continue until completion of the PCO cycle, albeit only processing PCO maintenance messages and discarding the other messages. Nodes that are instead in a maintenance phase, upon receiving a bootstrap message will not update their state, trying to avoid perturbing the collective clock attained at deployment.

2.4.1 Applications of Synchronization

The PCO protocol is a powerful primitive to achieve synchrony in networks of agents. As discussed in [85], synchrony is the ability to organize simultaneous

collective actions across the network, which does not imply a common notion of absolute time, or in other words, a global clock reference. However, synchrony is useful for many applications in the engineering arena; for example to coordinate collective actions in a distributed system. The mathematical models used to describe the collective behavior of fish schools and flocking inspired the development of synchronous linear and nonlinear models [42,62,63], used to update the velocity and the direction of each agent, in order to make them all equal, asymptotically. The PCO could pace correctly such updates. This type of synchrony may be useful also to achieve a TDMA in a decentralized fashion and in the context of cooperative communications. If a specific application, instead, requires a notion of absolute time, a reference needs to be flooded from a master node over the whole network.

2.5 Convergence in Complete Graphs

We introduce now the updating rule, which is a simplified version of the one discussed in [56], adapted in this context. Assume that one (or many) node fires at time t_k. The non-firing nodes update their phase variable as follows

$$\Phi_i(t_k^+) = \min\{(1+\alpha)\Phi(t_k), 1\}, \ \forall i \notin \mathcal{I}[k] \tag{2.5}$$

with $\alpha > 0$. If $\min\{(1+\alpha)\Phi(t), 1\} = 1$, a node is said to be *absorbed* and it will follow the same dynamics of the firing nodes. A single node can then be represented by a dot on a circle of unitary length (moving clockwise) at constant rate, whose initial position at time $t = 0$ is equal to $\phi_i(0)$ (see Figure 2.2).

Whenever a node crosses the finish line, i.e., its phase is equal to 1, the node emits a short pulse. The others, in turn, will push their phase forward getting closer to the firing node (see Figure 2.3). Nodes firing at the same time cannot hear each

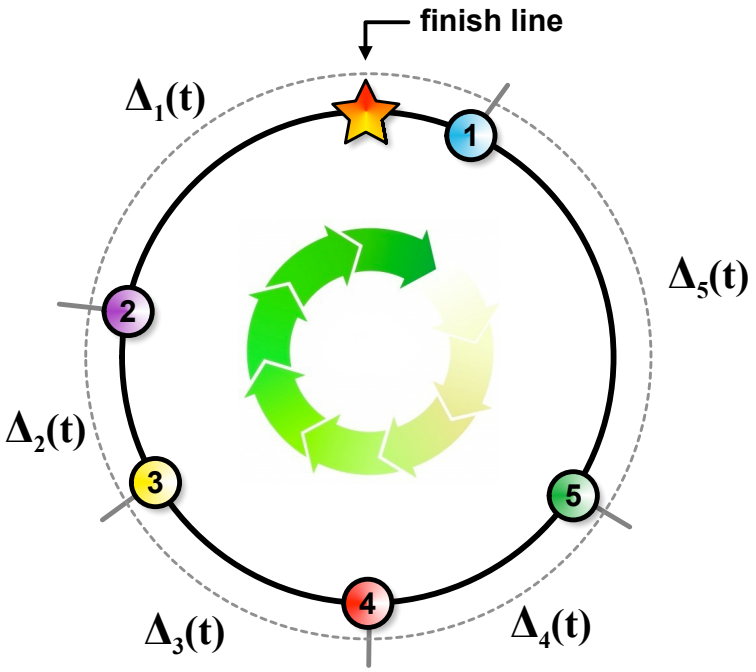

Figure 2.2: Network composed of 5 nodes. Each node moves clockwise. Without interactions, the distances between them remain constant.

other, due to the half-duplex constraint, and only the non-firing nodes will step forward. The idea is that if the nodes with a phase close to 1 make bigger phase jumps, the nodes will progressively tend to collapse, reaching synchronization. To simplify the study of the system, it is convenient to describe it in terms of phase differences. We define the *phase difference* between nodes i and $i+1$ at time t as

$$\Delta_i(t) = \Phi_i(t) - \Phi_{i+1}(t) \bmod 1 \tag{2.6}$$

which is illustrated in Figure 2.2 in the case of 5 nodes. As will be clear in the following, the updating rule is such that the order of firing times is preserved.

Without loss of generality, we assume that the nodes fire in increasing order of index, from 1 to n. Since no interactions take place when nobody is firing, we can look at the system at discrete times. In fact, if two consecutive firings happen to

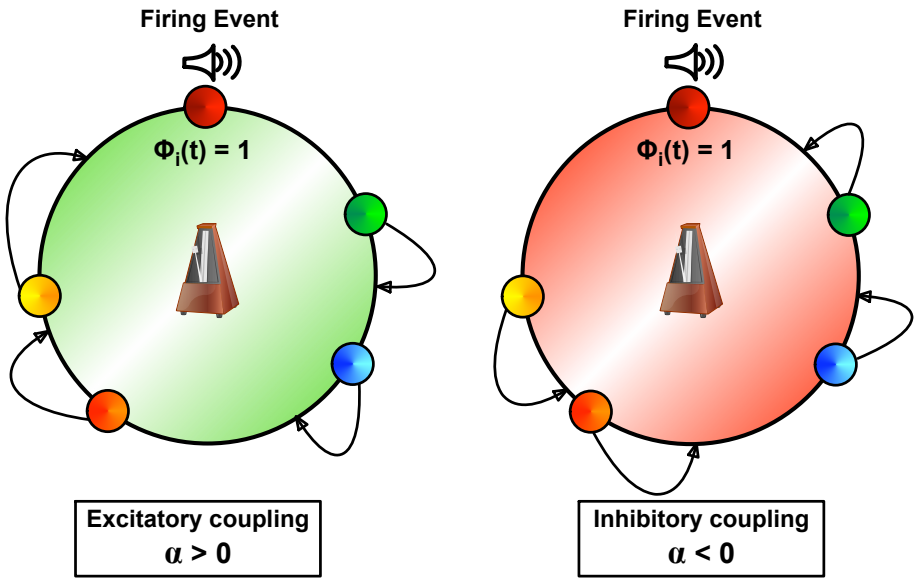

Figure 2.3: Left: PCO with excitatory coupling. Right: PCO with inhibitory coupling.

be at times t_1 and $t_2 > t_1$, then the phase differences between the nodes within the interval of time (t_1, t_2) do not change. Therefore, we can use and index k to count the firing events, and associate to each value of k the corresponding firing time t_k. We then indicate the set of nodes firing at iteration k as

$$\mathcal{I}[k] = \{i \in \mathcal{N} : \Phi_i(t_k) = 1\}.$$

The k^{th} firing event causes the non-firing nodes to update their phase variable, i.e.,

$$\Phi_i(t_k^+) = F(\Phi_i(t_k), \alpha), \ \forall i \in \mathcal{N} \setminus \mathcal{I}[k]$$

where α is a parameter, normally referred to as *coupling strength*. We further define a *round* as a set of n consecutive updates.[1]

We are going to show that this algorithm converges to synchrony, i.e., the dots

[1] In each round all nodes have fired once.

moving on the circle will progressively collapse into one single point. This implies that one of the Δ_is goes to 1 and the others go to 0. To simplify the geometrical interpretation, we assume that α is such that there are no absorptions in round R (we will explore later what happens otherwise). Consider a simple case, with $n = 3$ nodes, and suppose node 1 fires first round $R = 1$, at time t_1. At this time, $\Phi_1(t_1) = 1$. Because of the definition in (2.6) the phase differences are $\Delta_1(t_1) = 1 - \Phi_2(t_1)$, $\Delta_2(t_1) = \Phi_2(t_1) - \Phi_3(t_1)$ and $\Delta_3(t_1) = \Phi_3(t_1)$, respectively. Since $(1+\alpha)\Phi_i > \Phi_i, \forall \Phi_i \in (0,1)$ we have that

$$\begin{aligned}
\Delta_1(t_1^+) &= 1 - \Phi_2(t_1^+) = 1 - (1+\alpha)(1-\Delta(t_1)) \\
&= (1+\alpha)\Delta_1(t_1) - \alpha \\
\Delta_2(t_1^+) &= (1+\alpha)\Delta_2(t_1) \\
\Delta_3(t_1^+) &= (1+\alpha)\Delta_3(t_1)
\end{aligned}$$

At time t_2 node 2 fires, and the phase differences become

$$\begin{aligned}
\Delta_1(t_2^+) &= (1+\alpha)^2\Delta_1(t_1) - \alpha(1+\alpha) \\
\Delta_2(t_2^+) &= (1+\alpha)^2\Delta_2(t_1) - \alpha \\
\Delta_3(t_2^+) &= (1+\alpha)^2\Delta_3(t_1).
\end{aligned}$$

Finally, at time t_3 node 3 fires. At the beginning of round $R = 2$ we have that

$$\begin{aligned}
\Delta_1(t_3^+) &= (1+\alpha)^3\Delta_1(t_1) - \alpha(1+\alpha)^2 \\
\Delta_2(t_3^+) &= (1+\alpha)^3\Delta_2(t_1) - \alpha(1+\alpha) \\
\Delta_3(t_3^+) &= (1+\alpha)^3\Delta_3(t_1) - \alpha.
\end{aligned}$$

In the general case of n nodes, by repeating the procedure described above, it is possible to derive the phase differences at round $R+1$ as a function of those at round R as

$$\mathbf{\Delta}[R+1] = \mathbf{M}\mathbf{\Delta}[R] + \mathbf{v} \tag{2.7}$$

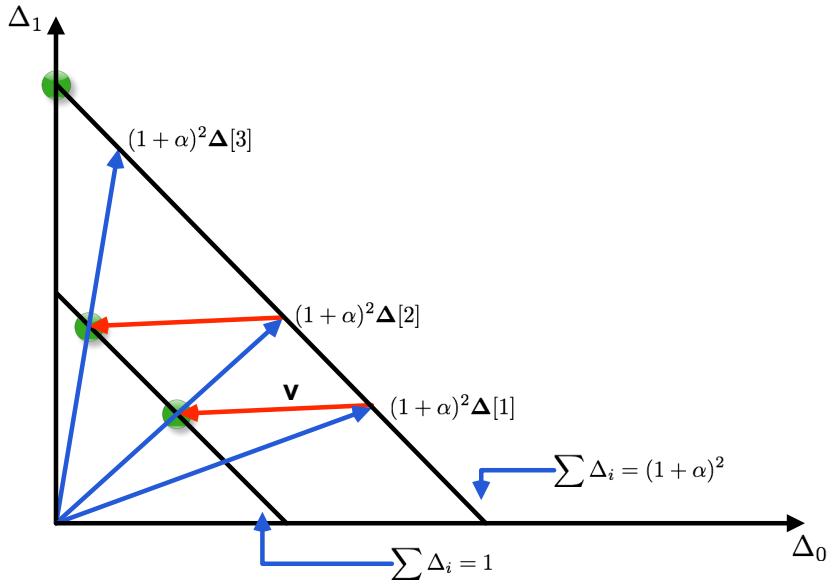

Figure 2.4: Geometric interpretation of PCO with excitatory coupling.

where $\mathbf{M} = (1+\alpha)^n\mathbf{I}$ and

$$\mathbf{v} = -\alpha[(1+\alpha)^{n-1}, \ldots, (1+\alpha), 1]^T.$$

It is clear from (2.7) that, by definition, $||\mathbf{\Delta}[R]||_1 = \sum \mathbf{\Delta}_i[R] = 1$ and, thus, $||(1+\alpha)^n\mathbf{\Delta}[R]||_1 = (1+\alpha)^n$. Since $0 \leq \Delta_i[R] \leq 1, \forall i$, $\mathbf{\Delta}[R]$ moves over the surface on the positive quadrant of R^n defined by the equation $\sum \Delta_i = 1$.

In Figure 2.4 is shown an example with 2 nodes: the system is specified by the vector $\mathbf{\Delta}[R] = (\Delta_1[R], \Delta_0[R])$. At round R, vector $\mathbf{\Delta}$ is amplified by a factor $(1+\alpha)^2$. Thus, $\mathbf{\Delta}$ is pushed over a surface of norm 1 equal to $(1+\alpha)^2$ and then a negative vector, $\mathbf{v}$, is subtracted. We can see that, since the vector $\mathbf{v}$ does not depend on $\mathbf{\Delta}[R]$, the result of the affine transformation $\mathbf{\Delta}[R+1] = \mathbf{M}\mathbf{\Delta}[R] + \mathbf{v}$ is a vector that constantly moves towards a fixed point. At round 2 the vector $\mathbf{v}$ will cause $\mathbf{\Delta}[3]$ to end up in a region where conditions $\sum \Delta_i = 1, 0 \leq \Delta_i \leq 1\ \forall i$ are not satisfied. This implies that an absorption occurs and the system synchronizes. If

$\mathbf{v}$ is such that its direction is exactly the same as $\mathbf{\Delta}[1]$, then the distances between the nodes will remain constant, but this happens with probability 0 over a set of random initial phases.

In the general case of n nodes, the same reasoning applies. Vector $\mathbf{\Delta}[R]$ is amplified by a factor $(1+\alpha)^n$ and added to $\mathbf{v}$. This makes $\mathbf{\Delta}[R]$ moving towards the borders of the region defined by equation $\sum \Delta_i = 1$. When $\mathbf{\Delta}[R]$ "hits" the border it means that an absorption occurred and, thus, the number of dots on the circle decreases. This explains why the phase difference between nodes n and 1 does not necessarily go to 1, as it would appear by assuming that no absorptions occur. If, for instance, node 1 is close to node n, it is possible for node 1 to be absorbed within a certain round, and some other Δ_i will converge to 1. This would considerably complicate the notation we previously used. However, the model in (2.7) gives geometrical explanation of why the system converges to a fixed point, which is the condition corresponding to synchronization. It is interesting to note that a negative coupling, i.e., $\varepsilon < 0$, will produce, instead, a completely different emerging pattern. In this case, in fact, the nodes will reach a condition where their firing separate in time of a constant amount, as we will see in the next Chapter.

2.6 Broadcast Networks

It is clear that firing at unison is a fixed point of the PCO protocol, because of the half-duplex constraint. In fact, if the nodes are synchronized, they all transmit the same message at the same time (or within a refractory period apart from each other), implying that they will not be able to receive while transmitting. Therefore, they will keep their clock unchanged and no interactions occur.

Let us denote with $t \in (0, \infty)$ the absolute time. We indicate with $\mathcal{G}(t)$ the topology of the network at time t. This choice allows us to handle switching topologies, as well as non idealities in the communication channel and the fact that transmission occurring exactly at the same time are not necessarily detected by other nodes. Specifically, assume that at time t some nodes, that we indicate with the set $\mathcal{I}(t)$, are transmitting a message. For any non firing node $j \notin \mathcal{I}(t)$ we define $0 < p_{j,\mathcal{I}(t)}(t) < 1$ as the probability of node j of detecting the message sent by the nodes in $\mathcal{I}(t)$. We assume that the network is connected in time: that is, $p_{j,\mathcal{I}(t)}(t)$ can be, occasionally, equal to zero, but every node has always a chance to hear messages from any other node in the network.

Since the current firing time is changed only when messages from the neighborhood are received, we use a discrete index $k \in (1, \infty)$ to denote the firing events, while $k = 0$ represents the random initial choice of firing times. The sequence of time instants t_k represents, then, the sequence of firing events as the time evolves (t_1 is the first firing event, t_2 the second and so on and so forth). We can, thus, define the set of firing nodes at iteration k as

$$\mathcal{I}(t_k) = \{i : \Phi_i(t_k) = 1\}. \tag{2.8}$$

Associated to $\mathcal{I}(t_k)$ we indicate with $\mathcal{U}(t_k) = \mathcal{N} \setminus \mathcal{I}(t_k)$ the set of nodes at time t_k that perform the update of their local state variable caused by the firing of the nodes within the set $\mathcal{I}(t_k)$.

The basic idea of the PCO protocol is that each node transmits its state to the neighborhood whenever its clock fires; therefore, the firing times encode the information about the state of each node, while also scheduling its transmission time and, therefore, regulating the exchange of information. Rather than considering phases or firing times, we associate each node to a color that represents its state.

Consider now the following game. There are n players, each with a randomly assigned color. The game stops whenever all nodes have the same color. The rules are simple. Each player wakes up according to its current color. For example, if the available colors are (yellow, orange, red, blue) we can assume that the nodes colored with yellow transmit first, followed by orange, red and blue.

Suppose a player, say i, wakes up. Any other player in the network, say j, whose color is not the same as i's, changes its own color to player i's one with probability $0 < p_{j,i}(t) < 1$. We call this procedure Coloring Game I. The convergence of this algorithm is established by the following theorem (the proof is in Appendix A.1).

Theorem 1 *The Coloring Game I converges almost surely.*

It is easy to see that the Coloring Game I is equivalent to the Strogatz model of the PCO with immediate absorption. In fact, the set of available colors is given by the initial (random) choice of firing times made by the nodes. Each firing causes some nodes to absorb others with some probability, and the absorbing state is the condition of synchrony. In the model proposed in [29] (immediate firing after absorption), instead, the existence of chains of absorbed nodes only modifies the transition probabilities $p_{j,\mathcal{I}(t_k)}(t_k)$. Specifically, this speeds-up its convergence properties, since repeated absorptions cause the set of synchronized node to increase. It is worth noticing the similarity between the Coloring Game I and the cascading behavior in Social Networks (see, for example, [40]).

Consider now the following game, that we call Coloring Game II. Each player chooses a color at random in a finite set of available colors. These colors are broadcasted, and the scheduling of transmissions is performed as before (each color has a corresponding transmission time). Every time a node receives a color

which is different by its own, the node modifies its state variable. In particular it moves towards it with some probability. The algorithm works as follows: the node tosses a coin and decide whether to perform the update or not. In the first case, if its color is too close to the one he just received, the node sets its state exactly equal to that, otherwise he moves towards it, choosing a color which is somewhere in the middle. In the latter case, the node keeps its own state. As opposed to Coloring Game I, new colors may appear in the process. The Coloring Game II is, basically, a random walk, where the state of the system is represented by the vector containing the colors of the nodes. At each iteration the new state of the system depends on the choices made by the nodes belonging to the set $\mathcal{U}(t_k)$, and these occur with probabilities that are defined by $p_{j,\mathcal{I}(t)}(t),\ \forall j, \mathcal{I}, t$. In practice, those values are unknown *a priori*, since they model mobility and channel non-idealities. However, in principle, the transition matrix $\mathbf{W}(t_k)$ from iteration k to $k+1$ exists and, thus, the state of the system $\mathbf{c}(k+1)$ at iteration $k+1$ is a function of $\mathbf{c}(k)$ through matrix $\mathbf{W}(t_k)$, that changes from iteration to iteration. The following results establishes the convergence of this algorithm (the proof is in Appendix A.2).

Theorem 2 *The Coloring Game II converges almost surely.*

Let us assume that the phase can only take discrete values, like little beans of a size equal to the refractory period, so that the maximum number of colors is finite.[2] The specifications of the Coloring Game II are equivalent, then, to the ones of the PCO protocol. In fact, the phase of a node or, equivalently, its next firing time, can be thought as a color that is exchanged with the neighbors. Each non firing node has a probability $p_{j,\mathcal{I}(t)}(t)$ of performing the update due to the firing

[2]This assumption is not restrictive, since clocks that differ less than a refractory period do not interact each other and, ultimately, any digital clock has a finite precision.

of the nodes in the set $\mathcal{I}(t)$, and the convergence of this procedure is guaranteed by Theorem 2. In particular, it follows that, under these assumptions, the PCO protocol converges in a mobile scenario. In fact, this latter case is equivalent to modify the detection probabilities of the nodes from iteration to iteration, and this does not change the convergence properties of our probabilistic interpretation.

As an example, in Figure 2.5 is reported the number of iterations required to achieve convergence as a function of the network size. Here, we considered the Strogatz model (no firing right after absorption), and, for simplicity, we assumed that the detection probability is the same for all nodes. We can see that, as the probability of detecting a message increases, the number of firings decreases exponentially. The significance of this result is as follows. The smooth approximation proposed in [50], which appears to lead to a model that does not ensure convergence in connected network [52], is inaccurate in predicting the synchronization conditions. Our proof shows that the ideal PCO must converge under milder conditions than those needed for the approximation proposed in [50] to converge, as conjectured and generally known from numerical examples to be true.

2.7 A software implementation of PCO

We describe now the software implementation of our synchronization protocol based on the PCO model. We first give a description of the possible events that need to be handled while performing the PCO protocol, deriving a simple protocol in pseudo−code. We then translate this layman description into an event-based implementation, suitable for an event-driven operating system, such as TinyOS. This includes a brief description of the μcontroller-radio interface, the radio man-

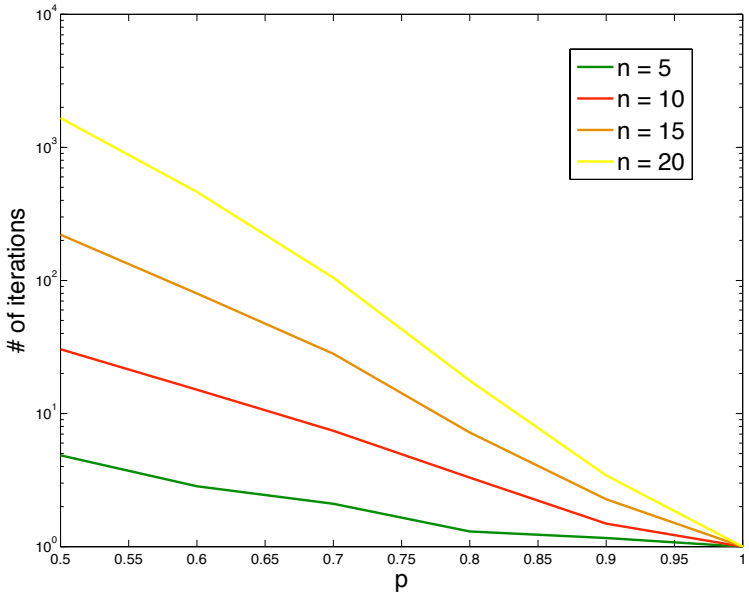

Figure 2.5: Number of firing events required to achieve convergence for different values of the network size.

agement, and how the two radio stacks (PCO and CSMA) are arbitrated within the microcontroller. Then, we introduce two possible versions of the PCO scheme, which roughly correspond to the two different cases considered in the PCO literature [29, 31, 56]. In the first version a node that is absorbed by another node, awaits for the next firing period to fire. In the second one, absorption events lead to immediate firing [29]. We call these two options PCOv1 and PCOv2, respectively. From an implementation point of view, they are almost identical, except for a few lines of code. To better illustrate the simplicity of the PCO scheme, in Algorithm 1 we report the pseudo-code in an event based fashion of the PCO protocol, suitable for software platforms currently available in the market, such as TinyOS.

Two possible events need to be handled while performing the PCO scheme: i) the internal clock firing, when the phase variable of node i reached the maximum

value $\Phi_i(t) = 1$, and ii) the detection of an incoming message. Both events are signaled by hardware interrupts that depend, in general, on the specific platform one uses. In our case, we implemented the PCO scheme on the MicaZ platform, composed of an Atmel128L (CPU) and a Chipcon2420 (radio). The **first** event (timer) is signaled by a timer overflow or an *output compare match*, interrupt. We made use of an output compare match interrupt on Timer3 (OCR3A), so that every time the counter of Timer3 (TCNT3) reaches a certain programmed value stored in the OCR3A register a hardware interrupt occurs and the interrupt service routine corresponding to the internal clock-firing takes place. In this case, the Timer3 counter is the local time, t_i = TCNT3, and the output compare value stored in register OCR3A is the period T = OCR3A. Hence, the normalized phase of the generic node i is simply given by $\Phi_i(t)$ = TCNT3/OCR3A. The Timer3 counter raises from TCNT3 = 0 to TCNT3 = OCR3A = T, at which time the node sends a message (i.e. *it fires*), as we will explain below, and resets the counter to the initial value TCNT3 = 0. The **second** event is the reception of a message, occurring whenever a preamble sequence has been detected. Due to the half duplex constraint, messages can be detected only if the node is not busy transmitting. As soon as a message has been detected the actual time should be corrected by subtracting the delay due to the packet length and the radio processing time. The total delay due to message handling and processing is given by $T_{\rm D} \simeq T_{\rm rx-tx} + T_{\rm cal} + 2T_{\rm pre-sfd} = 652\mu$s. T_D is the delay we used in Al.2. $T_{\rm rx-tx}$ is the period of time necessary to turn the circuitry from the receive mode to the transmit mode. $T_{\rm cal}$ is the frequency calibration time, and $T_{\rm pre-sfd}$ represents the time taken to transmit and receive the preamble sequence plus SFD byte. Note that only one of the two events described above can be managed by the micro-controller at once. Hence, the corresponding ISRs must disable all the interrupts

while running. In Figure 2.6 we provide a pictorial representation of the state machine of the PCO scheme. The local processor is constantly idle, unless either the clock fires or a message is received (which is captured by the SFD Interrupt). In the former case the next firing time is simply set as the current time plus one period. In the latter case, instead, if the local time is not within the refractory period, the clock is adjusted according to the updating function in Eq. (2.4). Algorithm 2 shows the pseudo-code in an event-driven fashion, which is a straightforward implementation of the state machine shown in Figure 2.6.

Algorithm 1: PCO Scheme

1. ***Event: Timer Fired***
2. set_timer(T);
3. send_msg();
4. ***Event: Message Received***
5. $\phi = t/T$
6. **if** $\phi < \phi_{\text{ref}}$
7. return;
8. **endif**
9. **if** $f(\phi) + \epsilon > 1$
10. set_timer(T);
11. send_msg();
12. return;
13. **endif**
14. $\phi = a_1\phi + a_0$
15. set_timer(T $- \phi$T)

As soon as the preamble sequence is detected at the micro-controller, the rest of the message does not matter, so we need this signal to be high for a sufficient

Algorithm 2: PCOv1 Algorithm

1. ***Initialization***
2. parameters setting;
3. Set_Clock(T);
4. ***Event: Input Capture Timer1***
5. **if** TCNT3 <= DELAY **then**
6. return;
7. **end if**
8. actual_time = TCNT3 − DELAY
9. **if** actual_time <= REF_PERIOD **then**
10. return;
11. **end if**
12. phase = actual_time/PERIOD
13. **if** phase >= phase_crit **then**
14. TCNT3 = 0
15. // send_msg() < − PCOv2 algorithm
16. return;
17. **end if**
18. phase = $a_1 * \text{phase} + a_0$
19. TCNT3 = phase ∗ PERIOD
20. ***Event: Timer3 Fired***
21. TCNT3 = 0
22. send_message();

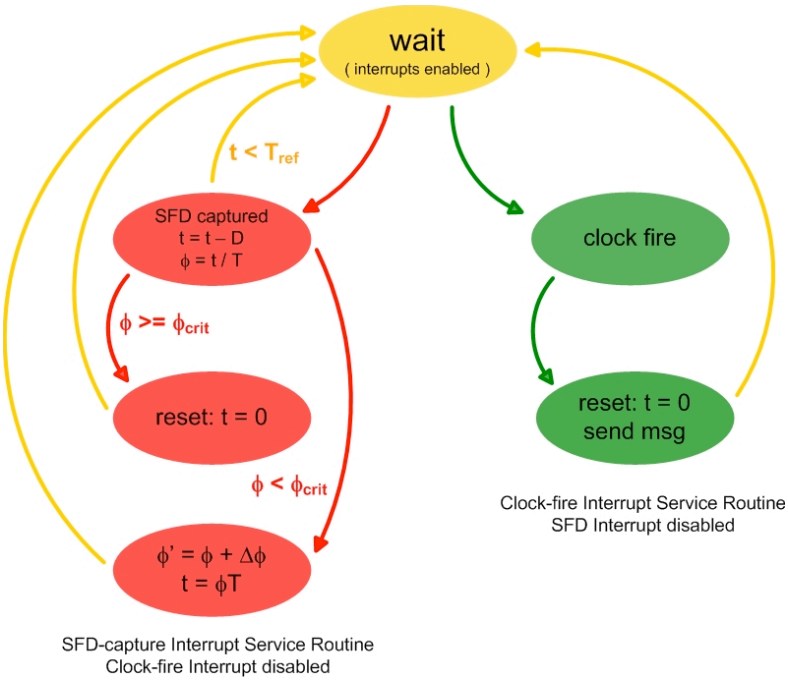

Figure 2.6: Finite state machine of the PCO algorithm.

amount of time needed to update the internal state variable, after which the receiver should turn immediately back to search for a new preamble. To achieve this, we used a common fake address composed by an invalid MAC sequence. In fact, if the automatic MAC address recognition is enabled, after the first comparison on the destination PAN field, the radio flushes the RXFIFO and starts to search for a new preamble.

Our code, being developed for the TinyOS environment, is portable, in principle, over any TinyOS sensor equipped with a CC2420 transceiver, with minor changes if the CPU is not an Atmel ATmega128L micro-controller. The hardware interface is shown in Figure 2.7. The upper, and most important part, is the MAC interface: the radio-chip informs the micro-controller of its activities through the

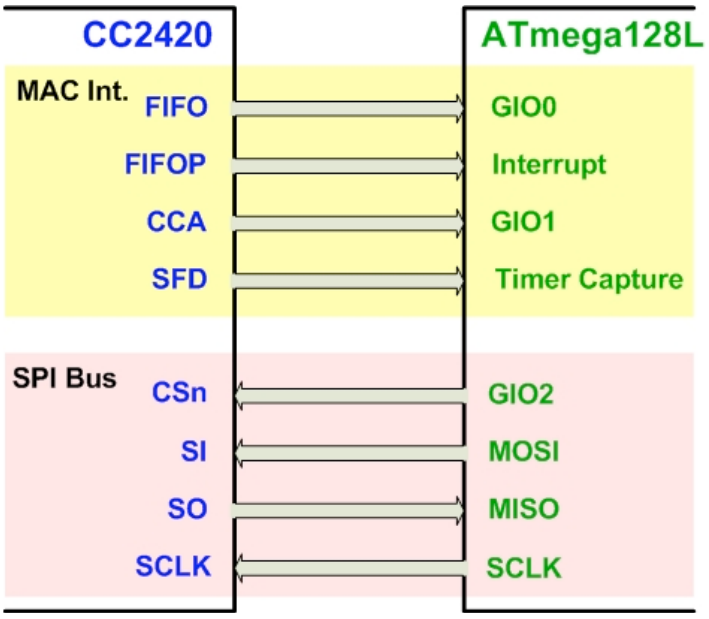

Figure 2.7: MicaZ SPI Bus and MAC Interface.

signals indicated. The detection of a message is signaled from the radio transceiver by a hardware interrupt on the SFD output pin, which is connected to the microcontroller input capture timer interrupt. We are able then to handle the detection of a signal through this interrupt.

The standard format of the MAC payload data unit is composed by a MAC address field (from 0 to 20 bytes), a payload and a frame control field (2 bytes). The MAC address field is composed, according to the standard version of TinyOS 1.x, of 9 bytes in total (Length [1], FCF [2], DSN [1], destination PAN [2], Address [2] and Group ID [1]). Before transmitting MAC and payload, the radio transceiver adds a preamble sequence (4 bytes) used to detect an incoming packet, followed by a start frame delimiter (SFD) byte, used by the receiver to lock the oscillator frequency. The message payload, plus a 2 bytes for frame check (FCS), are then appended.

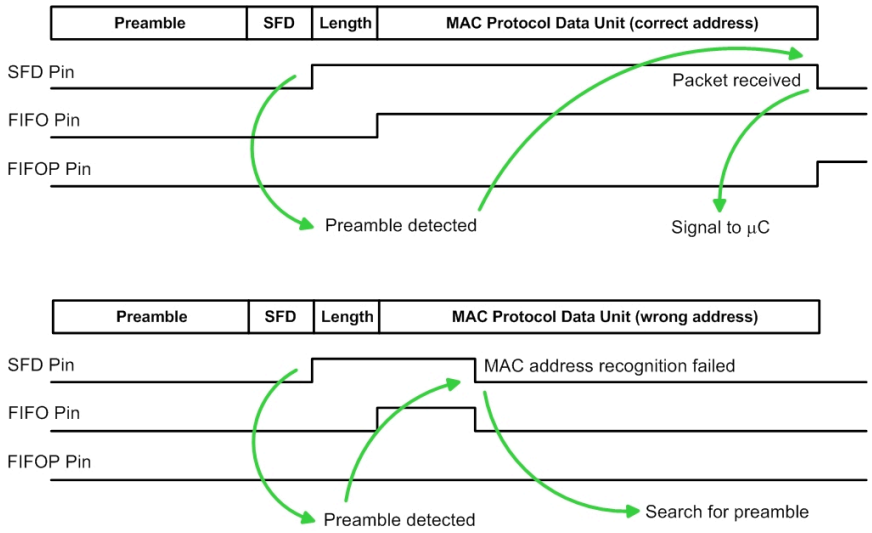

Figure 2.8: Radio Transceiver Receive Mode.

In receive mode, as the preamble and SFD fields are detected, the SFD Pin goes high. This signal is captured by the micro-controller, since the SFD Pin is directly connected to the Timer Input Capture Interrupt. If the address recognition is enabled, as in our case, it fails (as shown in the bottom part of Figure 2.8) because the address used by the PCO has been set, intentionally, to a non-existing address. Thus, the SFD and FIFO pins go immediately low. The received bytes are then automatically flushed, and the radio starts searching for another preamble. Figure 2.8 shows the transceiver receive mode in two cases: correct and incorrect address detection, respectively. The latter case (bottom) is used by the PCO scheme, in order to reduce the processing time of incoming messages, since no information other then the reception time is necessary to update.

2.7.1 The PCO layer

The CSMA-based protocol for medium access control is designed to avoid multiple transmissions in the same broadcast domain. The implementation in [85] works at the application layer, and does not need a low level interrupt handling to be performed, leaving the CSMA operations unaltered. A CSMA-based access scheme is not always the optimal choice for a distributed systems, as also discussed in [69]. In our particular case, large network or processing delays interfere with the PCO scheme, preventing the nodes from delivering and processing messages as soon as possible. This is the main reason that motivated us to handle directly hardware interrupts, splitting the network duty cycle in two part, synchronization and regular network activity. The added burden of handling the interrupts, as well as releasing a portion of time in each frame for synchronization purposes, is the price we have to pay to eliminate the network delays, as we explained in Section 2.4. One of the key differences in our implementation compared to [53] is the way we coerced the MAC layer to allow signals to coalesce and all nodes to transmit in unison, which is the aim of the PCO dynamics. As described in the following, we implemented an additional radio stack to support both the presence of the CSMA radio stack for regular data exchange and the PCO based radio for synchronization (obviously, only one of them can be used at a time, since the radio hardware is in common).

When a node transmits a packet, the data is first buffered into the radio transceiver. Then, the CSMA procedure starts: the micro-controller performs the RF power channel sensing through the radio ADC and, if the incoming power is less than a certain programmable threshold (usually set to -77dBm), the packet is delivered and the procedure terminates. If the medium is busy, the CSMA pro-

tocol repeats the steps described above for a fixed number of times (usually 8), and after which the packet is eventually discarded.

In contrast, the objective of the protocol is to make transmissions coalesce in time, which is exactly what the CSMA tries to avoid. While performing the synchronization protocol via PCO, the PCO radio stack has direct access to the radio-chip transceiver, through the cRadio module. Our implementation of the PCO primitive does not perform channel sensing and does not delay the transmission of the message, since additional delays could cause the algorithm not to converge.

Since the packet is the same for all the nodes during the synchronization, the common packet can be stored in the radio-transmission FIFO queue at the beginning of the algorithm and flushed at the end of it before turning the radio into the regular mode, resulting in a more efficient use of the hardware. A snapshot of this procedure is shown in Algorithm 3. The PCO algorithm runs for a fixed programmable number of iterations after which the `pcoDone` signal is sent to the main application and the TXFIFO is flushed so that regular packets can be buffered for regular transmissions. In Figure 2.9 a snapshot of the nesC style implementation of the architecture is shown.

2.7.2 PCOv1 and PCOv2

Two implementations of the PCO scheme are possible depending on how nodes behave in the face of *absorption*. Recall that a node is absorbed if upon detection of another node firing, the resulting state update makes the state variable greater than or equal to the threshold 1. **The first model** we considered, referred

Algorithm 3: delivery of a packet during the PCO algorithm

```
1. Command pcoRadioStatus.set(){
2.    *msg = common_packet;
3.    SFLUSH_TXFIFO;
4.    save(msg, TXFIFO);
5. }
6. // other code lines...
7. Event TXFIFO_DONE(){
8.    enable_pco_transmissions();
9. }
10. // other code lines...
11. Command pcoSend.send(){
12. if enabled then
13.       STXON; count++;
14.    if ++count==max_count then
15.       enabled = false; count= 0;
16.       FLUSH_TXFIFO;
17.       signal.pcoDone();
18.    end if
19. end if
20. }
```

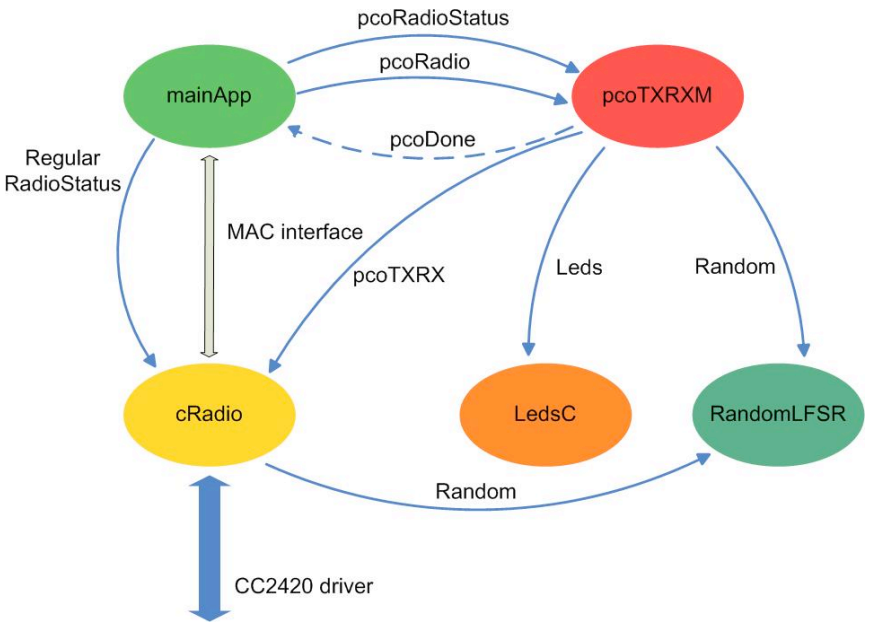

Figure 2.9: NesC style of the CSMA and PCO radio management.

to as PCOv1, is based on the assumption that an absorbed node waits for one cycle rather than firing immediately [31, 56]. In **the second model**, referred to as PCOv2, the absorption triggers an immediate firing event, i.e., an immediate delivery of a message [29]. This causes an avalanche effect that coerce the nodes to synchronize immediately. This implies a slight modification of the code, as shown in Algorithm 2.

To have a refractory period, we introduced an additional inactivity period $(T - \delta, T)$, within which the node does not update the phase variable due to an incoming message, where δ is of the order of the preamble length. The scheme implemented by PCOv2 requires a refractory period greater than the total round trip time of

the network. If there are many broadcast domains, it is not always possible to achieve this condition. Another possibility for avoiding protocol instability is to reduce the coupling strength. Scaling the coupling strength as the reciprocal of the average number of neighbors (roughly estimated) was proposed in [29] to minimize the energy spent (a case we tested in Section 2.8).

2.8 Experimental results

In this section we present the performance results of our protocol implementation of the PCO scheme and its comparison with the RBS protocol just described. The PCO algorithm drives all nodes to synchrony, hence, we would expect all of them pulsing at the same time after a sufficient number of iterations. We emulate the PCO bootstrap phase only.

To control the tests we used a single node in the network (the gateway) that sends a sufficiently strong signal to initialize the protocol. The nodes choose a random initial phase between 0 and T, where 0 is the time at which they received the start message from the gateway. The same node, once enough time elapsed so that with high likelihood all nodes have completed the preset number of firings, sends a data-acquisition signal, as shown in Figure 2.10 (top), to acquire the current firing times. The values received at the gateway are the firing times relative to the epoch at which the data-acquisition signal has been received. The accuracy with which the results are measured depends on this acquisition phase. Since the gateway is in range of all nodes, the difference between the time time of reception and the time of transmission is the small propagation delay difference that exists due to the different location of the recipients. The nodes indicate to the gateway how many residual tics of their local PCO clock, after the acquisition message reception, they

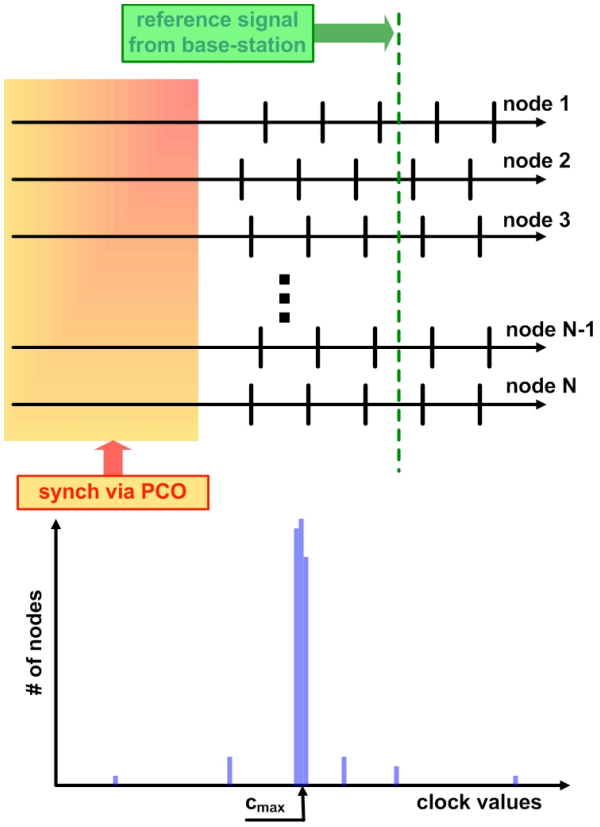

Figure 2.10: Data gathering, and derivation of $c_{\max}$.

plan to fire. Hence, the error in measuring the actual firing times has an additive term equal to the propagation delay, which is safe to consider negligible, plus a multiplicative term due to the clock offset that makes the time calculated by the node in terms of number of residual tics different from the actual time interval that the node will wait before firing. This error can be easily bounded because is in the order of the error that can be accumulated over at most a period. The base station then computes a histogram, in which the vertical axis is the number of node associated to a specific value of the clock. The bottom part of Figure 2.10 shows a typical histogram we would expect to see, where most of the nodes fall around an agreement clock time-stamp.

As indicated in Figure 2.10, we define $c_{\max}$ as the the central point of an interval

within which falls the majority of the nodes. Hence, $c_{\max}$ represents the value on agreement, while the difference in propagation delay can be considered negligible, since it is a 1-hop transmission over a relatively short range. The agreement value, $c_{\max}$, is derived as follows: each node has an internal clock, whose firing, of period T, is eventually synchronized with some other nodes within the network. If we scan all the clock readings from 0 up to T, using a sliding time window of size $\Delta << T$, we are able to define the number of nodes in agreement, n_{agree}, as the number of nodes for which the following quantity is maximized:

$$n_{\text{agree}} = \max_{\ell \in [0, T-\Delta]} \left\{ \sum_{i=1}^{N} u[c_i \geq \ell] u[c_i \leq \ell + \Delta] \right\} \tag{2.9}$$

where c_i is the clock of node i when the signal of the access point is sent after the PCO is done. Given ℓ_{agree} the argument that maximizes (2.9), the agreement value is calculated as

$$c_{\text{agree}} = \frac{\ell_{\text{agree}} + \Delta}{2}. \tag{2.10}$$

2.8.1 Comparison with RBS

Given the limitations of our testing environment we needed to validate the scalability of PCO by comparing it with other decentralized synchronization algorithms that run at the application layer. We chose the decentralized implementation of the RBS protocol for comparison. The first phase of this method requires the transmission of a reference signal after which all nodes record an estimated transmission time. The second phase uses *asynchronous average consensus* for generating consensus among the nodes on the estimated average transmission time of the reference signal. The consensus protocol is an iterative method characterized by a

very simple rule at each node-

$$x_i(t_i + 1) = x_i(t_i) + \alpha \sum_{j=1}^{N_i} (x_j(t_j) - x_i(t_i)) \tag{2.11}$$

which has been widely studied in the past [15,55,59,62,86]. Every time the internal clock of node i triggers a random update, the local variable x_i is changed according to the linear equation (2.11), in which the sum on the right side of the equation involves the differences between the values received by the neighbors and the local value weighted by a constant factor $\alpha = 1/d_{\max}$, where $d_{\max}$ is the maximum degree of the network[3]. Note that the comparison between PCO and RBS is not entirely fair, since RBS provides an absolute time reference, while PCO provides a time reference modulo the PCO period T. The consensus algorithm we implemented, given in Algorithm (4), is totally asynchronous and does not guarantee consensus to the exact average [8] but yields a much more rapid convergence to consensus compared to other alternatives [63]. Moreover, synchronous update require an additional cost due to synchronization, which requires, in turn, more energy to achieve this condition.

Each node has a single internal clock whose firing causes the emission of a packet containing its internal state variable. When the node is not transmitting, every incoming message is stored in a table whose values are used during the internal state update and then cleared.

[3]Different choices of the weights are possible [86]. Our choice is motivated by the fact that, given the knowledge (exact or approximated) of the topology, the rule $\alpha = 1/d_{\max}$ is very simple to implement, since all nodes use the same value. On the other hand, weights based on the local degrees require additional local information that requires an additional communication overload.

Algorithm 4: Asynchronous Average Consensus

1. *Initialization Phase*
2. read value;
3. Set_Clock(next_time);
4. neighbor_table ← [];
5. *Event Packet_Received:*
6. **if** found(neighbors_table) **then**
7. add_entry( );
8. **end if**
9. *Clock_Fired:*
10. **for** i = 1 : size(neighbor_table) **do**
11. **if** neighbor_table(i).received == 1 **then**
12. sum ← sum + const_weight · (neighbor_table(i).value - value);
13. **end if**
14. **end for**
15. value ← value + sum;
16. neighbors_table = [];
17. Send_Avg_msg(ID, value, local_degree);
18. Set_Clock(next_time);

2.8.2 Synchronization in Small Networks

We conducted experiments on both small and large topologies in order to verify the accuracy and the scalability of our implementation of the PCO protocol. In the topologies shown in the following, a link between two nodes exists if the percentage of received messages is greater than 80% in both ways (without interferers), and this has been determined experimentally. In the first set of of experiments we set $\epsilon = 1$, so that the reception of a message causes immediate absorption. As

an initial example, we tested a network with connectivity shown in Figure 2.11 where we have three broadcast domains connected to a central node (mote 4). The results for this topology are presented in Table 2.1, where PCOv1, PCOv2 and the distributed RBS are compared. For each experiment the nodes ran those protocols for a fixed number of iterations, ranging from 10 per node to 40 per node, and $\epsilon = 1$ (immediate absorption), averaged over 15 experiments for each case. The same table reports the number of firing groups, i.e., different groups of synchronized nodes, the standard deviation with respect to the agreed value (in the case of 2 firing groups the worst case is considered), the number of received messages per node, and the average total time to complete the algorithm. In these cases we set the coupling strength such that a firing caused immediate absorption of the neighbors. As we can see, the PCOv2 algorithm performs better than both the PCOv1 and distributed RBS protocols, and this can be explained with situations of stall that may arise in PCOv1 when nodes are absorbed by neighbors that are not in reach of each other, as the case we discussed in Section 2.7.2. The average standard deviation from the consensus value is low (ranging from 300μs to 400μs). Moreover, the number of times we experienced the formation of two different groups of synchronized nodes is smaller with respect to the PCOv2 protocol. This is due to the fact that when a node receives a message, and it is not within its refractory period, it immediately sends a message, hence, the information is sent to its neighbors right after the reception of the message. In other words, the diffusion of the information of the firing time is faster, and the formation of different groups of nodes firing at different times is less likely to happen. The number of received messages per node, when using the PCOv2 scheme is very small. This is due to the fact that as the nodes reach consensus, they cannot hear each other due to the half-duplex constraint. The PCOv2 scheme is very fast,

since the other two protocols are two/three times slower. The high variance of the synchronization time of the PCOv1 protocol is due to the fact that there is no delivery of message due to absorption. Hence, many absorbed nodes might remain absorbed for long periods, while others complete their number of firings relatively soon.

Table 2.2 shows the performance results for the topology depicted in Figure 2.12. In this case we have 24 motes and the resulting connectivity map is composed by different broadcast domains and nodes 7-8 act as bridges between long distance nodes. As we can observe, the formation of two synchronized groups is more frequent for the PCOv1 protocol, while the PCOv2 scheme always leads to global synchronization as the number of iteration increases. In this last case, the best synchronization accuracy is of the order of 300μs, while the time taken to reach synchronization ranges from 20 to 32 periods, with 30 and 40 firings per node respectively. As observed so far, the PCOv2 scheme is still the best in terms of speed of convergence and accuracy. In [85] a similar topology was considered: the coupling strength ranged from $\epsilon = 1/100$ to $\epsilon = 1/1000$, while the time-to-synch ranged from 284.3s to 1164.4s, with a period of $T = 1$s. The system was able to synchronize within 410.4μs in 284.3s with a coupling strength of 1/100, while a better standard deviation in the final error was reached with $\epsilon = 1/1000$. In this latter case, however, the system took 1164.4s to synchronize.

In our case, as seen in Table 2.2, a similar network was able to synchronize (all nodes within the same firing group) in 10s to 40s, with a standard deviation of the order of a few hundreds of micro-seconds. Hence, the accuracy of the PCOv1 and PCOv2 implementations is of the same order of the accuracy obtained in [85], but in our case the time taken to synchronization is one order of magnitude smaller.

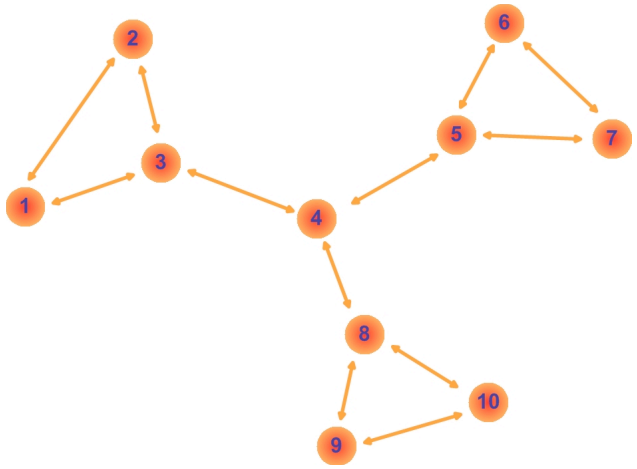

Figure 2.11: Topology 1: connectivity map of a network composed by 10 nodes.

Table 2.3 shows the performance results for the topology depicted in Figure 2.13, referred to a network composed by 50 nodes. As we can observe, the increasing number of nodes does not have a significant impact on the performance of the PCO protocols, since they present speed of convergence of the same order of the previous cases. On the other hand, the distributed implementation of the RBS protocol is much slower in this case, because the network has doubled in size. Again, the fact that more nodes are within the same network does not represent a problem for the PCO protocols, in particular the PCOv2 scheme, since the superposition of signals is used to enforce the diffusion of information.

Table 2.1: Performance results of PCOv1, PCOv2 and Distributed RBS, with period $T = 1$s, transmission power $P_{\text{tx}} = -10$dBm, referred to topology of Figure 2.11.

PCOv1

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	2 (80%)	1700	14 ± 10	16 ± 5
20	2 (75%)	900	26 ± 15	35 ± 10
30	2 (50%)	552	30 ± 20	55 ± 15
40	2 (30%)	425	35 ± 22	75 ± 20

PCOv2

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	2 (30%)	400	5 ± 2	5 ± 1
20	2 (20%)	390	9 ± 3	14 ± 2
30	2 (20%)	390	14 ± 4	20 ± 2
40	2 (10%)	300	18 ± 5	29 ± 2

Distributed RBS

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10		15	24 ± 4	11 ± 1
20		13	51 ± 20	21 ± 1
30		12	82 ± 20	32 ± 1
40		3	111 ± 50	42 ± 1

2.8.3 Synchronization in Large Networks

During our experiments, we noticed the formation of synchronized groups only at the lowest power level, -25dBm. As the power was increased this phenomenon quickly disappeared, especially within the range $(-10\text{dBm}, 0\text{dBm})$. Hence, we decided to count small groups of synchronized nodes as nodes not synchronized. Note that this performance metric is less optimistic than the one used within the previous section. Given the agreed clock value in Equation (2.10), we define the outage probability P_{out} as the percentage of nodes outside a specified interval, say Δ_T, around c_{agree}. Hence, $1 - P_{\text{out}}$ is the percentage of nodes whose clock value is such that $|c_i - c_{\text{agree}}| < \Delta_T/2$. Figure 2.14 shows the percentage of nodes $1 - P_{\text{out}}$ within an interval of time $\sim \Delta_T = 500\mu$s.

The coupling strength used in PCOv2 was set to $\epsilon = 1/d_{\text{avg}}$, where d_{avg} is the average degree per node within the network. This choice has two advantages: 1) it helps the stability of the PCO scheme in the sense that jitters do not cause a node to align with them since ϵ is small enough; 2) it causes an avalanche effect in PCOv2, as discussed in [29], which in turn makes its speed of convergence faster with respect to the PCOv1 and the RBS protocol. In PCOv1 we set the coupling strength equal to $\epsilon = 0.1$ in all experiments within this section. In fact, in this case absorbed nodes do not transmit a message immediately. Therefore, $\epsilon = 1/d_{\text{avg}}$ does not cause the avalanche effect. Our choice was based on preliminary experimental results that indicated similar performance within the range $\epsilon = (0.1, 0.4)$.

As we can see from Figure 2.14, PCOv1 and PCOv2 have the same performance except at a transmission power level of -25dBm in which case the PCOv2 performs much better because of the avalanche effect. Note that PCOv1 and PCOv2 have almost the same performance with 10 and 50 iterations per node. This is due

Table 2.2: Performance results of PCOv1, PCOv2 and Distributed RBS, with period $T = 1$s, transmission power $P_{\text{tx}} = -10$dBm, referred to topology of Figure 2.12.

PCOv1

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	2 (70%)	3500	32 ± 15	29 ± 6
20	2 (60%)	875	44 ± 20	40 ± 10
30	2 (50%)	600	45 ± 20	71 ± 15
40	2 (20%)	520	50 ± 21	87 ± 15

PCOv2

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	1	320	6 ± 1	7 ± 1
20	1	300	13 ± 4	9 ± 2
30	1	290	15 ± 3	20 ± 2
40	1	290	19 ± 4	32 ± 2

Distributed RBS

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10		45	49 ± 16	13 ± 1
20		36	92 ± 22	24 ± 1
30		32	152 ± 43	35 ± 2
40		30	183 ± 21	44 ± 2

Table 2.3: Performance results of PCOv1, PCOv2 and Distributed RBS, with period $T = 1$s, transmission power $P_{\text{tx}} = -10$dBm, referred to topology of Figure 2.13.

PCOv1

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	2 (40%)	4100	28 ± 12	29 ± 6
20	2 (30%)	1600	48 ± 20	41 ± 12
30	2 (30%)	1280	72 ± 29	60 ± 13
40	2 (25%)	1050	86 ± 34	82 ± 14

PCOv2

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10	2 (50%)	420	10 ± 1	5 ± 1
20	2 (20%)	400	13 ± 2	11 ± 2
30	2 (10%)	400	16 ± 4	14 ± 2
40	1	290	21 ± 5	19 ± 2

Distributed RBS

# of firings	# of groups	std dev [μs]	# of msg	total time [s]
10		128	59 ± 7	15 ± 1
20		72	105 ± 11	26 ± 1
30		59	172 ± 18	36 ± 2
40		32	223 ± 19	45 ± 2

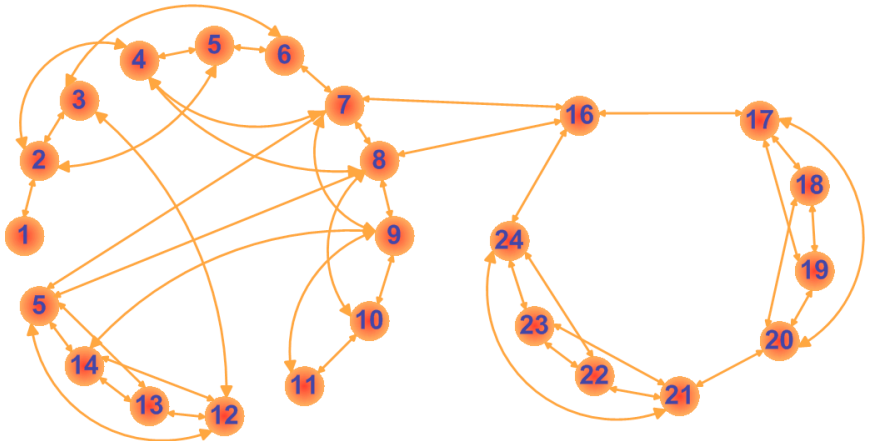

Figure 2.12: Connectivity map of a network composed of 24 motes. The performance results are shown in Table 2.2.

in part to the relatively high average number of neighbors and in part due to the fact that the power of many synchronized nodes adds-up, so that they can reach longer distances. On the other hand the consensus based synchronization protocol performs well only with 50 iterations per node, while with 10 iterations the convergence seems not to happen. Moreover, at high transmission power levels loss of messages due to congestion is experienced. In Figure 2.15, for example, we report the number of received messages per node versus the transmit power per node, for a fixed number of iterations equal to 50, for PCOv2, PCOv1, and for RBS with average consensus. Note that, the number of messages received per node is quite low for both PCOv2 and PCOv1, and this can be explained considering the fact that when the nodes become synchronized they cannot hear each other. On

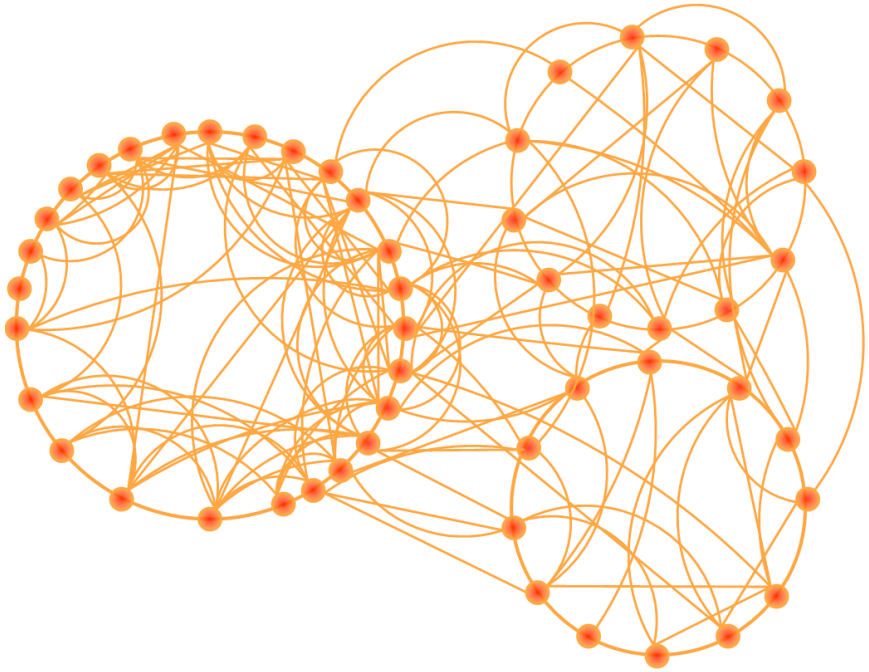

Figure 2.13: Connectivity map of a network composed of 50 motes. The performance results are shown in Table 2.3.

the other hand, the distributed version of the RBS protocol experiences many more messages received per node, which increases from -25dBm to -15dBm; interestingly, the trend is not monotonic and the number of messages received suddenly decreases when the power exceeds -15dBm. This can be explained with packet losses that are due to network congestion, caused by reduced spectral reuse.

As we can see from the results we obtained, the PCOv2, corresponding to the version of the PCO scheme with immediate absorption, performs better than PCOv1 and RBS respectively. PCOv2 with 10 firings per node has the same per-

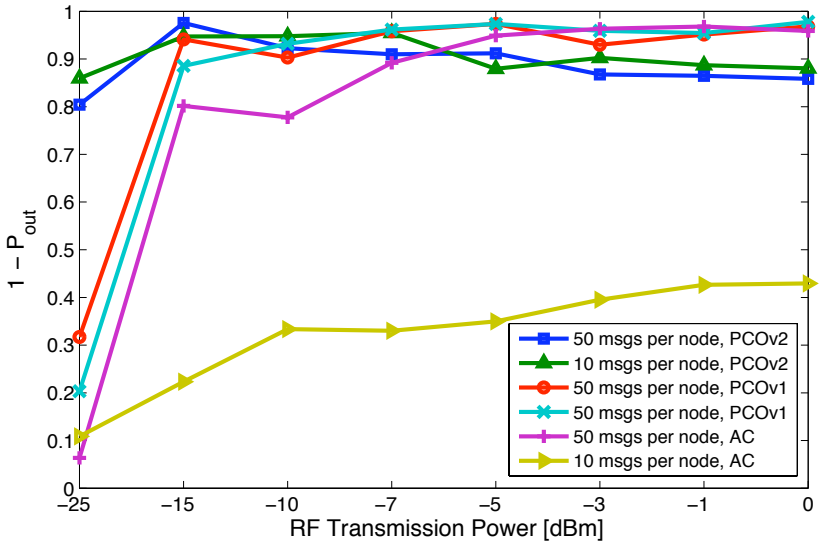

Figure 2.14: PCOv1, PCOv2 and AC: Accuracy as a function of the transmission power ($N = 100$, $T = 50$ms) (each value has been averaged over 15 to 20 experiments. Coupling strength: [0.010 (0dBm), 0.012(-1dBm), 0.013(-3dBm), 0.014(-5dBm), 0.017(-7dBm), 0.025(-10dBm), 0.040(-15dBm), 0.050(-25dBm)]).

formance of PCOv1 with 50 firings per node. Moreover, at low power transmission levels [-25dBm, -15dBm], PCOv1 is unable to reach synchronization either with 10 or 50 firings per node. From the same Figure we can see that the decentralized RBS protocol performs no better than the PCOv1. Hence, we conclude that PCOv2 offers the best performance in terms of accuracy. Furthermore, from the results we reported regarding Small Networks, it is evident that the PCOv2 scheme has the smallest time required to achieve synchronization, and exhibits a smaller probability of causing the formation of different synchronized groups. The PCOv2 protocol offers better performance with respect to the PCOv1 scheme because in the first case, the absorbed node emits immediately a message, eventually reaching other nodes. In other words the information diffuse much faster over the network via PCOv1. This is shown by our results, since they show that the PCOv2, with

10 firing per nodes, achieves the same level of accuracy of PCOv1, with 50 firings per node.

Remarks

The experiments were conducted in simple scenarios and the performance results presented above are obtained by averaging each case over 15 to 40 experiments. Obviously this is not sufficient to prove that the real performance of the specific implementation of our protocol corresponds to the truth for several reasons, such as the impossibility to run experiments in many scenarios, the limited available time-resource to gather more data, the difficulties encountered for the time synchronization offset, and the data gathering itself. Moreover, a refractory period has been introduced within the PCO scheme, so that the standard deviation on the final accuracy of the PCO protocols is guaranteed to be within the refractory period we set to 2ms, but it could be different numerically, provided that the differences between any two nodes within the network fall within 2ms. An additional problem was that we could not connect each mote directly to a computer in order to record step by step the evolution of the network state. Hence, the final result is affected by an error which is the sum of two terms: one due to the PCO protocol and another one due to the clock drift. If the typical clock drift is 100ppm, then the maximum difference of any pair of clock frequencies is given by $2\Delta f = 1474$Hz at $f_0 = 7.3728$MHz. Roughly speaking, an estimate of the maximum drift per second is then given by $f_0\,(1/(f_0 - \Delta f) - 1/(f_0 + \Delta f)) = 200\mu$s. This is a worst case scenario, but from this estimate we can see that the gathering of data must be accomplished as soon as possible in order to reduce the worse case clock spread due to the drift. This error may then affects significantly the measurements of

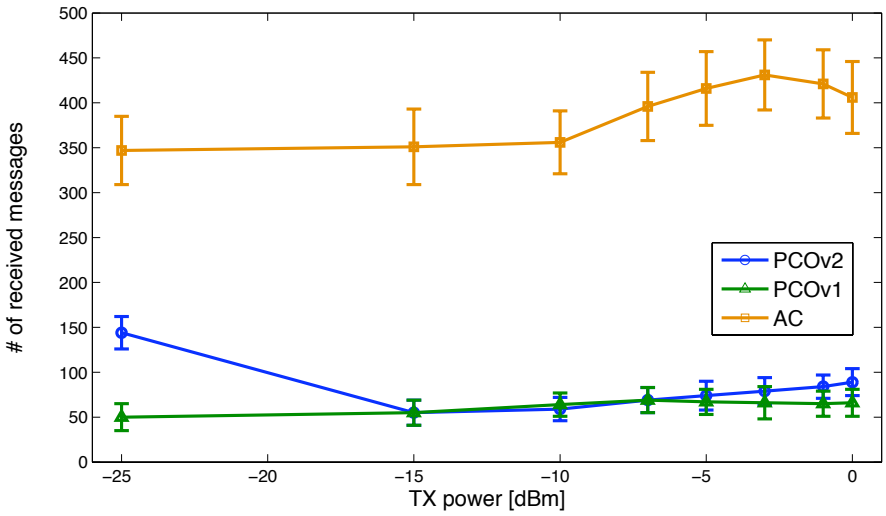

Figure 2.15: PCOv1, PCOv2 and AC: Number of messages received per node, as a function of transmission power and 50 iterations per node (each value has been averaged over 40 experiments).

the PCOv1 protocol, since the finishing time of nodes varies significantly. On the other hand, the convergence time of the PCOv2 protocol has a small variance, so that those data are not too much corrupted by other sources of error. Moreover, we can see that for small-size networks, the formation of two synchronized groups occurs with some probability. This is due in part to non-idealities in both the hardware interface of the motes and the physical layer, and in part to interference with other wireless networks, since we tested our protocols during the day, and they cannot be avoided. However, we can see that if the number of iteration is sufficiently large, that probability quickly decreases, and that this phenomenon is less frequent for large networks, such as 100 nodes, for example. This effect could be further mitigated by making a node change its own phase whenever it realizes that its neighbors are divided into two synchronized groups.

CHAPTER 3

DECENTRALIZED PRIMITIVES FOR TIME-DIVISION MULTIPLE ACCESS IN WIRELESS BODY AREA NETWORKS

3.1 Motivation and Related Work

In the previous chapter we focused our attention on PCO as a primitive for synchronization and discussed aspects ranging from its convergence properties to its practical implementation on commercial radio. In Section 2.5 we discussed the possibility of inverting the coupling signal sign and mentioned that it leads to a result that is complementary to that of PCO: specifically, the pulses tend to separate each other by a fixed amount. This fact, well known in the mathematical biology literature was first proposed as a mechanism to enforce a TDMA schedule in [19]. This chapter describes new protocols based on the same basic idea that are appropriate for very low power sensor applications. We target the application of Body Area Networks, where the deployment is most suited to apply these concepts.

The design of multiple access for Wireless Body Area Networks (WBAN), should account for two basic application requirements: i) the sampling of biological signals occurs regularly, making rather inefficient the use of asynchronous Carrier Sensing Multiple Access (CSMA) protocols that are adopted in most commercial interfaces, such as for examples, 802.11, Bluetooth, and 802.15.4 [77]; ii) the signals are samples at heterogeneous rates that depend on the nature of the phenomenon under observation [37] (for example, motion sensors, ECG sensors, breathing sensors, blood glucose sensors etc.). The number of sensors used in different applications vary and may range from a handful of sensors to a few dozens.

Thus, in this context one would want a multiple access control (MAC) primitive capable of generating a regular Time Division Multiple Access (TDMA) schedule with a possibly variable number of nodes and satisfying heterogeneous bandwidth demands. Even if WBANs have naturally a clustered structure, decentralized MAC protocols are often lightweight and adaptive. The advantage of CSMA is precisely its decentralized nature, that makes it suitable in applications where nodes are intermittent, the size of the network is unknown and the traffic demands are unequal. However, for periodically sampled data, TDMA is more efficient and there is no reason to employ an asynchronous medium access policy, if TDMA can be achieved in a simple decentralized fashion.

As observed by several Authors (see e.g. [77]), the specifications of Zigbee [2], Bluetooth [1], Mica Motes [32] and other commercial platforms are orders of magnitude above the limits imposed by many WBAN applications. In general, engineering WBANs can considerably benefit from employing an integrated system point of view in designing network primitives, based on novel and simple physical models for the communications interactions and for the computation of the network state.

In the literature, a number of papers has already been devoted to promoting the use of PCO as a synchronization primitive [29, 50, 82, 85]. By inverting the polarity of the coupling signal, it has been observed in [56] that the PCO network can be led to a pulsing pattern that yields constant spacing between neighboring pulses. This emergent behavior, called *desynchronization*, has been utilized in [68] to achieve round-robin scheduling, where each node in the network is allotted an equal share of the total transmission time. Specifically, in [68], it has been shown that a consensus-like updating rule produces uniform spacing in $O(n^2)$ iterations.

In this Chapter, we first show that the original PCO model with inhibitory coupling yields only weak desynchronization, which refers to the case where a constant spacing between consecutive nodes' pulsing times is reached but the differences between their local phase variables shift constantly over time. Although round-robin scheduling is achieved in this case, the convergence is not robust to non-ideal effects of the environment since it relies on the constant interaction between PCOs. Then, by restricting each pulse coupling to only a subset of nodes, we show that it is possible to achieve strict desynchronization, where the phase differences among the nodes also remain constant over time. Given knowledge of the number of nodes n, the primitive we describe is shown to achieve desynchronization with a faster rate than the scheme proposed in [68]. More interestingly, we also show that, by having each node maintain two local clocks instead of one, we can further achieve proportional fair scheduling where the time allotted to different nodes are proportional to their demands.

Building on such bio-inspired primitive, we consider the UWB ON-OFF PCO radio design in [5] as our canvas, and complete it with the necessary primitives for access and two way communications with the access point of the WBAN. Once again, to give scalable designs, we propose to harmonize active and inactive transmission and reception times across the network with very simple dynamics inspired by nature.

3.2 Round-Robin Scheduling with PCO Desynchronization

The problem of desynchronization can be considered the dual of the synchronization problem, where the states of the nodes coalesce under the action of the network dynamics. Instead, nodes that are desynchronized will alternate their pulses in order with constant spacing between each other. The constant spacing produced by the PCO dynamics can be utilized as the basis of round-robin scheduling [19]. That is, each node can utilize the time period between its own firing and the firing of the next node to transmit its data. Specifically, suppose that the nodes are indexed in an increasing order of their initial phases such that $\phi_1 < \phi_2 < \cdots < \phi_n$ (i.e., the nodes pulse in the inverse order of their indices). As we show later on, this firing order will be preserved by all the schemes proposed in this work. When each node fires exactly once, from node n to node 1, we say that we have completed a *round* of firing. As in the case of the classic PCO model, the phase of a node evolves according to

$$\Phi_i = \frac{t}{T} + \phi_i \bmod 1$$

Again, we can imagine the set of n nodes as balls placed on a circle of circumference equal to 1 (the normalized period) moving clockwise at the same speed in decreasing order of their indices. Whenever a node, say node i, crosses the finish line (i.e., $\Phi_i(t) = 1$), it emits a pulse that triggers the others to update their local phase variables. The difference with what we studied in the previous sections lies in the updating function that will be introduced below.

As done in the case of excitatory coupling, to study the evolution of the system, we keep track of the phase differences between two consecutively firing nodes

change over time, i.e.,

$$\Delta_i(t) = \Phi_i(t) - \Phi_{i-1}(t) \pmod 1, \tag{3.1}$$

with

$$\Delta_1(t) = \Phi_1(t) - \Phi_n(t) \pmod 1. \tag{3.2}$$

3.3 Notation and Definitions

Each firing modifies the phase differences between nodes and, thus, the state of the system can be described by the vector

$$\mathbf{\Delta}(t) = (\Delta_n(t), \ldots, \Delta_1(t))^T$$

which follows a trajectory over the n-dimensional plane defined by $\sum_{i=1}^{n} \Delta_i(t) = 1$, subject to the constraints that $\Delta_i(t) \in (0, 1)$, $\forall i$. We can achieve two types of convergence, namely, strict and weak desynchronization, as described below.

Definition 1 *The network is strictly desynchronized if the phase differences converge to a constant value as $t \to \infty$. This is equivalent to saying that, for any $\epsilon > 0$, there exists t^* such that*

$$|\Delta_i(t) - c_{\text{strict}}| < \epsilon \tag{3.3}$$

for all $t > t^*$ and $i = 1, \ldots, n$.

When strict desynchronization is reached, the phase difference between any two consecutive nodes converges to the same fixed value and, thus, the spacing between two consecutive firings also remains constant. However, to achieve equal spacing

between nodes' firing times, it is actually not necessary to have the phase differences remain constant over time. In fact, the phase differences can shift frequently over time as long as the time between consecutive firings remain constant. This leads to the notion of weak desynchronization as described in the following.

Recall that the sequence of n consecutive firings starting at node n and ending at node 1 constitutes a *round* of firing. Let us denote by $t_i^{(k)}$ the firing time of node i in the k-th round. By definition, a round terminates after the firing of node 1, which occurs at time $t_1^{(k)}$ for round k. Therefore, we define the phase differences at the end of round k as

$$\Delta_i[k] \triangleq \lim_{\eta \downarrow 0} \Delta(t_1^{(k)} + \eta), \quad \text{for } i = 1, \ldots, n \tag{3.4}$$

and the state of the system in round k as

$$\mathbf{\Delta}[k] = (\Delta_n[k], \ldots, \Delta_1[k])^{\mathrm{T}}. \tag{3.5}$$

It is worth noticing that $\Delta_1[k]$ is the difference in firing time between node 1 and node n, which is the next one to fire. Since any node can be labeled as node 1 by a shift of the time of origin, the convergence of $\Delta_1[k]$ to a fixed value can be viewed as the convergence of firing time between any two consecutive nodes.

Definition 2 *The network is weakly desynchronized if the time elapsed between consecutive firings converges to a constant value, i.e., for any $\epsilon > 0$, there exists k^* such that*

$$|\Delta_1[k] - c_{\text{weak}}| < \epsilon \tag{3.6}$$

for all $k > k^*$ and $i = 1, \ldots, n$.

When strict desynchronization is reached, the nodes will fire at fixed time instants in each period. When weak desynchronization is reached, instead, the time

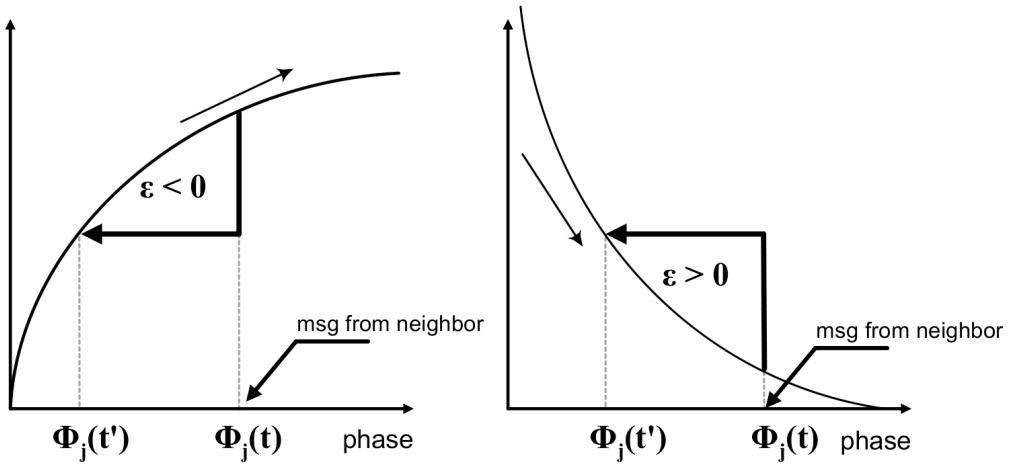

Figure 3.1: Left: Strogatz model with concave-down f and $\varepsilon < 0$. A message received from the neighborhood causes the node to step back in phase (its next firing time increases). Right: PCO-based model with concave-up f and $\varepsilon > 0$. This is a similar model to the one shown on the left. After detecting a message, the node add a positive quantity to its state, but since f is concave-up, its phase decreases.

between two consecutive firings converges to a constant, but the phase differences are not fixed for all t. Clearly, strict desynchronization implies weak desynchronization, but not the other way around.

Definition 3 *The convergence time R^* is the minimum number of rounds needed to achieve ϵ-desynchronization, i.e., convergence with ϵ accuracy.*

3.4 Weak Desynchronization

The protocol we describe first is inspired by the PCO model, introduced by Peskin [70], with inhibitory coupling. Rather than changing the polarity of the coupling, we maintain $\varepsilon > 0$ and introduce a convex function f (see Figure 3.1). This system produces the same global effects as described in the following. Specifically, by taking the dynamics as

$$f(\Phi_j(t)) = -\log(\Phi_j(t)) \tag{3.7}$$

and positive coupling $\varepsilon = -\log(1-\alpha)$, where $\alpha \in (0,1)$, the firing of node i at time t will trigger node j to update its phase as

$$\begin{aligned} \Phi_j(t^+) &= f^{-1}(f(\Phi_j(t)) - \log(1-\alpha)) \\ &= (1-\alpha)\Phi_j(t). \end{aligned} \tag{3.8}$$

We can now state the following result (see Appendix B.1 for proof).

Theorem 3 (Weak Desynchronization) *For any $\alpha \in (0,1)$, the dynamics in (3.8) will reach weak desynchronization with $c_{\text{weak}} = \frac{\alpha}{1-(1-\alpha)^n}$, except over a set of initial conditions of measure zero. The convergence occurs in $R^* = \left\lceil -\frac{1}{n}\frac{\log\left(\frac{2}{\epsilon}\right)}{\log(1-\alpha)} \right\rceil$ rounds.*

Although the time required to achieve weak desynchronization scales favorably as $O(\frac{1}{n})$ with respect to the network size, the algorithm itself is not robust to detection errors as to be shown in the following. This is due to the fact that the phase differences between nodes do not remain constant over time and, thus, achieving equal spacing between consecutive firing times must rely on constant interaction among the nodes. This motivates the study on strict desynchronization.

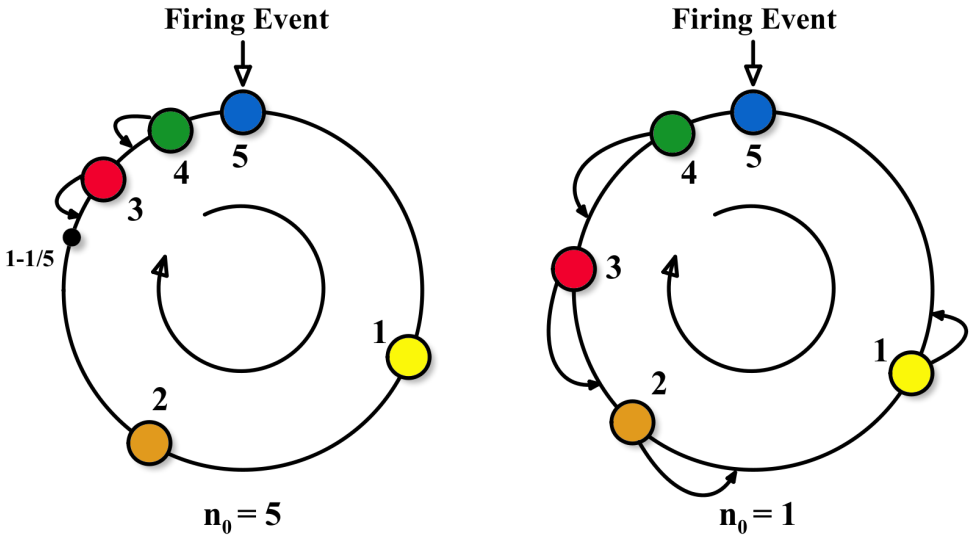

Figure 3.2: Update in PCO-Based Round-Robin, with $n = 5$ nodes. Left: $n_0 = n$; in this case only the nodes whose phase is bigger than $1 - \frac{1}{5}$ shift back in phase. Right: $n_0 = 1$; a firing causes all the others to change their phase.

3.5 Strict Desynchronization

One of the reasons why the previous scheme can only achieve weak desynchronization is because the firing of each node will always impose a coupling on all the others. Thus, the nodes phases will be constantly updated throughout each round. To achieve strict desynchronization, our intuition tells us that one should restrict the coupling caused by each firing event to only a subset of neighboring nodes. That is, the firing of a node should only push away the nodes whose phase is too close. Based on this intuition, we introduce the following updating rule (assuming

node i fires at time t)

$$\Phi_j(t^+) = \begin{cases} f^{-1}(f(\Phi_j(t)) + \varepsilon), & \text{if } \Phi_j(t) \in \left(1 - \frac{1}{n_0}, 1\right) \\ \Phi_j(t), & \text{otherwise.} \end{cases}$$

$\forall j \neq i$, where the parameter $n_0 \geq 1$ controls the spacing between nodes, and the dynamics

$$f(\Phi_j(t)) = -\log\left(1 - n_0(1 - \Phi_j(t))\right) \tag{3.9}$$

is a concave-up function similar to that in (3.7) and in Figure 3.1 but defined only over the range $\left(1 - \frac{1}{n_0}, 1\right)$. We can see that the phase of node j is pushed toward the point $1 - 1/n_0$ through a convex combination with parameter $\alpha = 1 - e^{-\varepsilon} \in (0, 1)$. In fact,

$$\begin{aligned} \Phi_j(t^+) &= f^{-1}(f(\Phi_j(t)) + \varepsilon) \\ &= (1 - \alpha)\,\Phi_j(t) + \alpha\left(1 - \frac{1}{n_0}\right), \end{aligned} \tag{3.10}$$

for all $j \neq i$ such that $\Phi_j(t_i) \in (1 - \frac{1}{n_0}, 1)$. It is interesting to note that (3.9) reduces to (3.8) when $n_0 = 1$. We can now state the the following results (see Appendix B.2 for proof):

Theorem 4 (Strict Desynchronization) *For any $\alpha \in (0, 1)$, the dynamics in* (3.9) *with $n_0 = n$ will reach strict desynchronization for all initial conditions, except over a set of measure zero.*

By assuming that the firing of each node imposes a coupling only on the node immediately firing after it (which is typically the case when the nodes are close to the desynchronous state), we show in Appendix B.3 for a special case that strict desynchronization can be achieved in $R^* = O(\frac{n}{\log n})$ number of rounds. The algorithm we described reaches strict desynchronization in a way that depends on

the number of active nodes. Therefore, extra complexity needs to be introduced to set the parameter n_0 appropriately.

On the other hand, in the DESYNCH method proposed in [68], each node accepts coupling only from one node (i.e., the node firing immediately after it), regardless of the number of nodes in the network. Therefore, the knowledge of n is not required. Specifically, upon hearing the pulse emitted by node $i-1$, node i will update its local phase $\Phi_i(t)$ by deliberately moving it towards the middle-point between the phases of nodes $i+1$ and $i-1$, which is the time that node i should be firing if desynchronization is reached (i.e., the target phase). That is, by assuming that node $i-1$ fires at time t_{i-1}, node i will update its phase to

$$\Phi_i(t_{i-1}^+) = (1-\alpha)\Phi_i(t_{i-1}) + \alpha\Phi_i^{\text{target}}(t_{i-1}^+), \tag{3.11}$$

where $\alpha \in (0,1)$ is the coupling parameter and $\Phi_i^{\text{target}}(t_{i-1}^+) = \frac{\Phi_{i+1}(t_{i-1}^+)+\Phi_{i-1}(t_{i-1}^+)}{2}$ is the target phase.[1] It is necessary to note that node i can only attain knowledge of the phases of nodes $i-1$ and $i+1$ by observing their pulsing times. By assuming that t_{i+1} is the most recent firing time of node $i+1$, the estimate that node i has on the phase of node $i+1$ is

$$\hat{\Phi}_{i+1}(t) = \Phi_i(t) + [1 - \Phi_i(t_{i+1})],$$

where $1-\Phi_i(t_{i+1})$ is the phase difference between nodes $i+1$ and i at the time node $i+1$ was firing. However, the phase of node $i+1$ would have shifted by the time node $i-1$ is firing (due to the firing of node i) and, thus, would no longer be accurate when node i updates its local phase variable. A conjecture on the convergence of the DESYNCH protocol was given in [19] and is restated below.

Claim 1 ([19]) *n nodes governed by* DESYNC *will always be driven to strict desynchronization (for $n < 500$) in $R^* = \frac{1}{\alpha}n^2\log\left(\frac{1}{\epsilon}\right)$ rounds.*

[1] Recall that $\Phi_{i-1}(t_{i-1}^+) = 0$ since a node resets its phase after emitting a pulse.

3.6 Discussion

The different protocols we have discussed so far for round-robin scheduling belong to the same class of algorithms that use time of transmission to encode the state of each node and use pulse coupling as the basis of nodes' interaction. The basic PCO model with negative coupling provides convergence in a *weak* sense, which is still useful to schedule the nodes' duty-cycles, but is not robust to pulse transmission errors since the convergence relies on constant shifting of phases triggered by the pulse transmissions. When coupling of each pulse is limited to only a selected group of nodes, as in the case of the PCO-based scheme with thresholding or DESYNCH, strict desynchronization can be achieved.

Although the PCO-based scheme and DESYNCH both belong to the class of pulse-coupling algorithms, they have inherent differences. In DESYNCH, each node must identify the nodes firing immediately before and after it and deliberately record an estimate of their phases. In the PCO-based schemes, each node can accept a coupling whenever the relative time difference between the reception of a pulse and the local firing fall within a certain interval set by the threshold. There is no need for the PCO-based schemes to keep track of the adjacent nodes and their phases. However, the disadvantage of the PCO scheme lies in the fact that knowledge of the network size n is required in order to guarantee equal-share scheduling. Nevertheless, convergence to a fixed schedule can always be achieved even when n_0 is not equal to n, but the schedule may not result in equal share and the type of convergence will depend on the value of n_0 (e.g., weak for $n_0 < n$ and strict for $n_0 \geq n$). More importantly, the convergence time of DESYNCH is $O(n^2)$ while PCO with $n_0 = n$ requires only $O(\frac{n}{\log n})$.

In this section, we have shown how equal-share scheduling can be achieved

by having nodes follow a simple pulse-coupling mechanism. In the following section, we shall extend the ideas of pulse coupling to the case of proportional fair scheduling by using more than one clocks at each node.

3.7 PCO Primitives for Proportional Fair Scheduling

In many networking applications, different nodes may have different bandwidth requirements and, thus, should be allocated unequal shares of the transmission time that are proportional to their demands[2]. In this context, we assume that each node has a specific request that is quantified by the value K_i, where i is the node index. The value K_i lies between 1 and a maximum value $K_{\max}$, which is fixed *a priori* and equal for all the nodes, that is, $K_i \in [1, K_{\max}]$ with $K_{\max} < \infty$. Basically, K_i stands for the amount of resources node i is hoping to obtain. Since our objective is to derive a negotiation scheme not based on explicit exchange of information, the nodes may be requesting more than the actual available resource. Hence, our goal is to have each node obtain, by negotiation, a portion of time equal to $\frac{K_i}{K} T_f$, where $K = \sum_{i=1}^{n} K_i$ is the total demand of all nodes.

Following the formulation in the previous section, an intuitive approach is to have each node, say node i, maintain K_i virtual nodes that employ the PCO-based scheme with the threshold mechanism or DESYNCH. Since each virtual node will obtain one out of K shares of the transmission time T_f, node i will obtain a total K_i/K portion of the transmission time. However, this approach is obviously inefficient since each node needs to emit K_i pulses in each cycle and the convergence time also increases proportionally. In the following, we show that proportional fair

[2]An example of proportionally fair schedule is in the protocol IS-895, aimed at unleashing multi-user diversity taking into consideration fairness issues in the channel assignment.

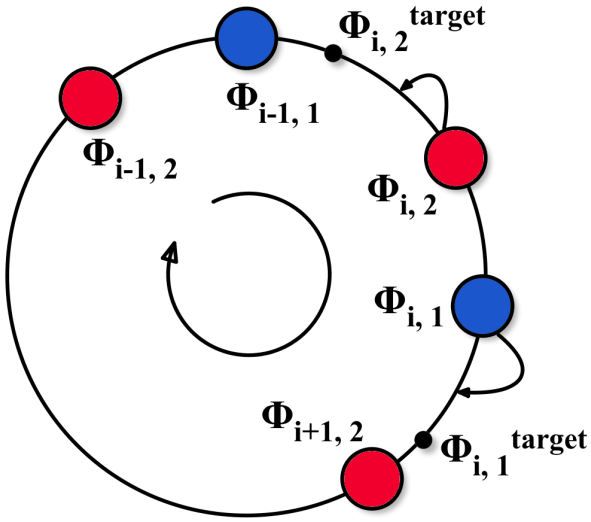

Figure 3.3: Proportional Fairness: update of node i, caused by the firing of node $i-1$. Node i either expands or stretches its phase variables, depending on its neighbors' states. The objective of node i is given by $\Phi_{i,1}^{\text{target}}$ and $\Phi_{i,2}^{\text{target}}$.

scheduling can actually be achieved by having each node maintain only *two* clocks and by embedding the demand K_i in the update rule. The basic idea is to treat each node as a spring of force with constant K_i. If we take many springs, each of which obeys the Hooke's law $F_i = -K_i x_i$ with x_i being the difference between the actual and natural lengths of spring i, and connect them together in a circle, then each spring will occupy a portion of the circle that is proportional to K_i/K. We can then imagine n users having different requests K_i as n springs connected together in a ring. The two pulses emitted by each node mark the boundaries of the spring on the circle and the update rule incorporates the strength of the spring that is parameterized by K_i. Each user needs then to communicate only with its two neighbors, implying that only two state variables are truly needed.

Each node has two phase variables, indicated by $\Phi_{i,1}(t)$ and $\Phi_{i,2}(t)$, where i is the index of the node. Let $\phi_i \in [0,1)$ be the initial phase of both phase variables of node i. When nobody is firing, the two phases will evolve as

$$\begin{aligned}\Phi_{i,1}(t) &= (\phi_i + t/T_f) \bmod 1,\\ \Phi_{i,2}(t) &= (\phi_i + t/T_f) \bmod 1,\end{aligned}$$

and fire when either $\Phi_{i,1}(t)$ or $\Phi_{i,2}(t)$ reaches the value 1. Our goal is to expand the time between the firings of the two clocks at each node (with $\Phi_{i,1}(t)$ firing before $\Phi_{i,2}(t)$ as illustrated in Figure 3.3) and use it as the transmission period scheduled for that node. By assuming that an extra δ portion of the unit share (defined as $1/K$ of the total time scheduled for the nodes' transmission) is used as guard band between the transmission periods of adjacently transmitting users, the network will require a total of $K+n\delta$ shares and node i should ideally obtain $\frac{K_i}{K+n\delta}$ portion of the transmission time when proportional fair scheduling is achieved. In that case, the target phases of node i should be

$$\begin{cases}\Phi_{i,1}^{\text{target}}(t) = \frac{\delta}{K_i+2\delta}\Phi_{i-1,1}(t) + \frac{K_i+\delta}{K_i+2\delta}\Phi_{i+1,2}(t)\\ \Phi_{i,2}^{\text{target}}(t) = \frac{K_i+\delta}{K_i+2\delta}\Phi_{i-1,1}(t) + \frac{\delta}{K_i+2\delta}\Phi_{i+1,2}(t).\end{cases} \tag{3.12}$$

In the proposed scheme, we allow node i to update the phases of its two local clocks immediately upon the firing of the first clock of node $i-1$. The update is ideally performed by moving the local phases towards the target such that

$$\begin{cases}\Phi_{i,1}(t_{i-1,1}^+) = (1-\alpha)\Phi_{i,1}(t_{i-1,1}) + \alpha\Phi_{i,1}^{\text{target}}(t_{i-1,1}^+)\\ \Phi_{i,2}(t_{i-1,1}^+) = (1-\alpha)\Phi_{i,2}(t_{i-1,1}) + \alpha\Phi_{i,2}^{\text{target}}(t_{i-1,1}^+)\end{cases} \tag{3.13}$$

where $\alpha \in (0,1)$ and $t_{i-1,1}$ is the firing time of the first clock of node $i-1$. The update is illustrated in Figure 3.3. Recall that, at time $t_{i-1,1}^+$, we have $\Phi_{i-1,1}(t_{i-1,1}^+) = 0$ and, thus, the target phases reduce to $\Phi_{i,1}^{\text{target}}(t_{i-1,1}^+) = \frac{K_i+\delta}{K_i+2\delta}\Phi_{i+1,2}(t_{i-1,1}^+)$ and $\Phi_{i,2}^{\text{target}}(t_{i-1,1}^+) = \frac{\delta}{K_i+2\delta}\Phi_{i+1,2}(t_{i-1,1}^+)$.

The following result establishes the convergence of this idealized version of the updating procedure (see Appendix B.4 for proof).

Theorem 5 [PROPORTIONAL FAIRNESS] *The algorithm in (3.13) converges for all initial conditions* $(\phi_1, \phi_2, \ldots, \phi_n)$*, except over a set of measure zero. Furthermore, by letting* $\Gamma_i(t) = \Phi_{i,1}(t) - \Phi_{i,2}(t) \pmod 1$*, and* $\Theta_i(t) = \Phi_{i,2}(t) - \Phi_{i-1,1}(t) \pmod 1$*, we have*

$$\lim_{t\to\infty} \Gamma_i(t) = \beta\frac{K_i}{K} \quad \textit{and} \quad \lim_{t\to\infty} \Theta_i(t) = \beta\frac{\delta}{K},$$

for all i*, where* $\beta = \frac{1}{1+\frac{n\delta}{K}}$ *and* $K = \sum_{i=1}^{n} K_i$*.*

The theorem indicates that the portion of time node i obtains converges to βK_i, where β is the unit share. When the guard interval is made arbitrarily small, that is, as δ goes to 0, node i will obtain exactly K_i/K portion of the firing cycle T_f for transmission. However, it is necessary in practice to have $\delta > 0$ not only to accommodate the synchronization errors and non-negligible pulse widths, but also to leave a vacant time window for external nodes to join the network. In any case, proportional fair scheduling is achieved since the time alloted to the users are indeed proportional to their demands.

This approach is similar in nature to the DESYNCH and thus has the same problem regarding the estimate of the phase of the node firing before it. Therefore, we can similarly replace the phase $\Phi_{i+1,2}(t^+_{i-1,1})$ in the update rule with the phase estimate

$$\hat{\Phi}_{i+1,2}(t) = \Phi_{i,1}(t) + [1 - \Phi_{i,1}(t_{i+1,2})],$$

where $t_{i+1,2}$ is the most recent firing time of the second clock of node $i+1$. This estimate is obtained by having node i observe the firing time of node $i+1$ and

compute the relative phase difference between itself and node $i+1$ at the time of firing. However, since the estimate of the phase of node $i+1$ is no longer accurate by the time node $i-1$ is firing, it is possible that pulsing order between clocks $\Phi_{i,1}(t)$ and $\Phi_{i+1,2}(t)$ may exchange positions after the update. That is, the period in between the two pulses of nodes i and $i+1$ may overlap. To avoid this problem, we modify the target values defined in (3.12) as follows

$$\begin{cases} \tilde{\Phi}_{i,1}^{\text{target}}(t) = \min\left(\Phi_{i,1}^{\text{target}}(t), \frac{\Phi_{i,1}(t)+\Psi_{i+1,2}(t)}{2}\right), \\ \tilde{\Phi}_{i,2}^{\text{target}}(t) = \max\left(\Phi_{i,2}^{\text{target}}(t), \frac{\Phi_{i-1,1}(t)+\Phi_{i,2}(t)}{2})\right). \end{cases} \tag{3.14}$$

This choice avoids the possibility that the clocks interleave their phases. As discussed in Appendix B.5, the fixed point is preserved and no other fixed points exist. The dynamics in (3.13) with target values defined by (3.14) is similar to the ideal case, with the only difference that the update is limited by max and min to avoid overlapping. Numerical simulations show that the convergence properties are not affected.

3.8 Results: Part I

In this section we provide and discuss the performances of the algorithms we discussed for achieving round-robin scheduling and proportional fairness based on PCO-like primitives.

Round-Robin Scheduling

In Figure 3.4, we show the evolution of the phase distances $\Delta_i[k]$ according to the dynamics specified in (3.9) with $n = 5$ and coupling parameter $\alpha = 0.25,\ 0.5,\ 0.9$.

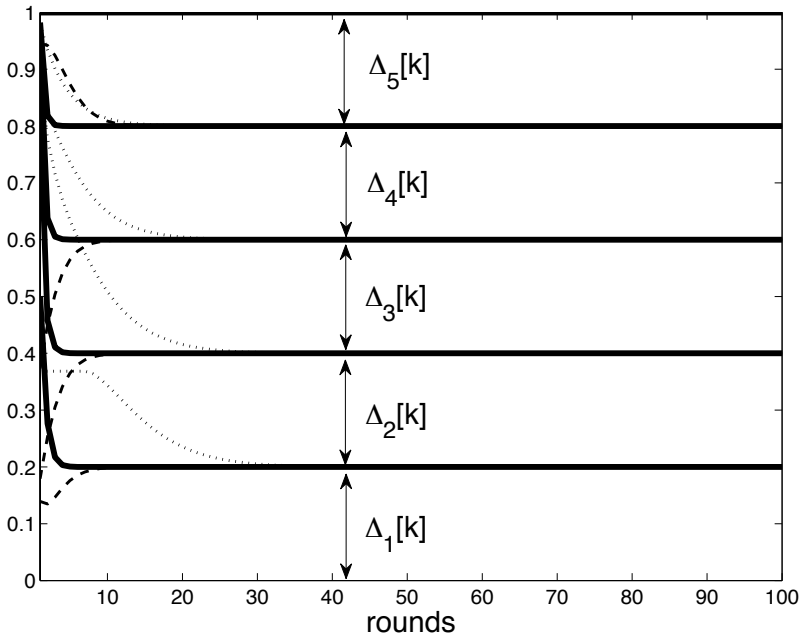

Figure 3.4: Evolution of phase distances with $n = 5$ nodes, $n_0 = n$ and different values of α ($\alpha = 0.25$ [dotted], $\alpha = 0.5$ [dashed], $\alpha = 0.9$ [solid]).

The difference between the curves from top to bottom are $\Delta_5[k]$, $\Delta_4[k]$, ..., $\Delta_1[k]$, as indicated in the figure. Since all phase differences converge to $1/n = 0.2$, we can infer that strict desynchronization is achieved (see Section 3.3). Also, as expected, increasing the coupling strength results in a faster convergence to the fixed point $\frac{1}{n}\mathbf{1}$.

In Figure 3.5, we show the evolution of the system state for a network with $n = 5$ nodes and parameters $\alpha = 0.4$ and $n_0 = 3 < n$. This is an illustration of weak desynchronization which occurs whenever $n_0 < n$. Recall that the dynamics described in (3.7) is a special case of (3.9) with $n_0 = 1 < n$. We observe that the phase differences do not converge to the same value, but the difference in consecutive firing times remains fixed as indicated in Figure 3.6, which shows the

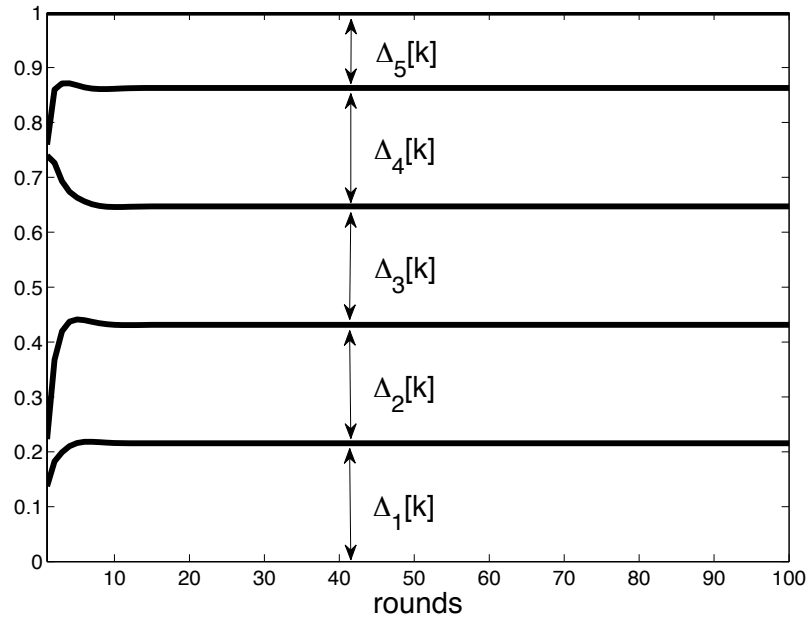

Figure 3.5: Evolution of a network with $n = 5$ nodes, $n_0 = 3$ and $\alpha = 0.4$. We can see that the phase differences do not converge to the same value.

difference in firing time with respect to the firing events.

In Figure 3.7, we reported the number of rounds R required to achieve desynchronization, with $\epsilon = 10^{-4}$. We can see that the DESYNCH protocol exhibits a complexity that scales as $O(n^2)$, as conjectured in [19]. The PCO-based dynamics in (3.9) exhibits sublinear complexity, while in the case of weak desynchronization it is of the order of $\frac{1}{n}$.

Now, let us consider the non-ideal effects of the transmission channel. We consider the case where each node experiences miss detection with probability p each time a pulse is emitted. Assume that the miss detection probability p is equal for all nodes and that signal-to-noise ratio is sufficiently high such that false alarm is negligible[3]. In Figure 3.8, we reported an example of weak desynchronization

[3]The issues regarding processing and propagation delays are not considered explicitly in this

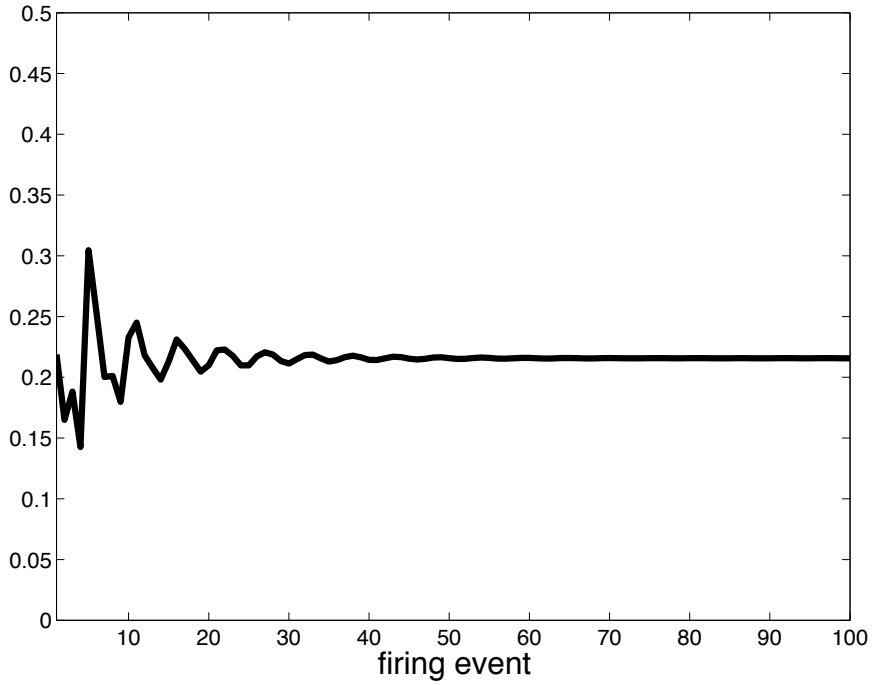

Figure 3.6: Time between consecutive firings for the PCO-based algorithm, with $n = 5$, $n_0 = 3$ and $\alpha = 0.4$. Although the phase differences do not converge to the same value, the time elapsed between consecutive firings converges.

with $n = 10$ nodes and $\alpha = 0.1$ in two different cases: the ideal case and the case with miss detection probability $p = 0.1$. We can see that the desynchronous state is not steadily maintained in the case of weak desynchronization since it relies on constant interaction between PCOs, which can be disturbed by detection errors. In the case of strict desynchronization, as shown in Figure 3.9, the desynchronous state can be better maintained since the interaction between PCOs are not required once the this state is reached. However, the time needed to reach the convergent state is now increased.

work since we are dealing with a network with closely located nodes, such as BAN. However, these issues can be treated by employing refractory periods as done in [29], but may affect the precision of the schedule.

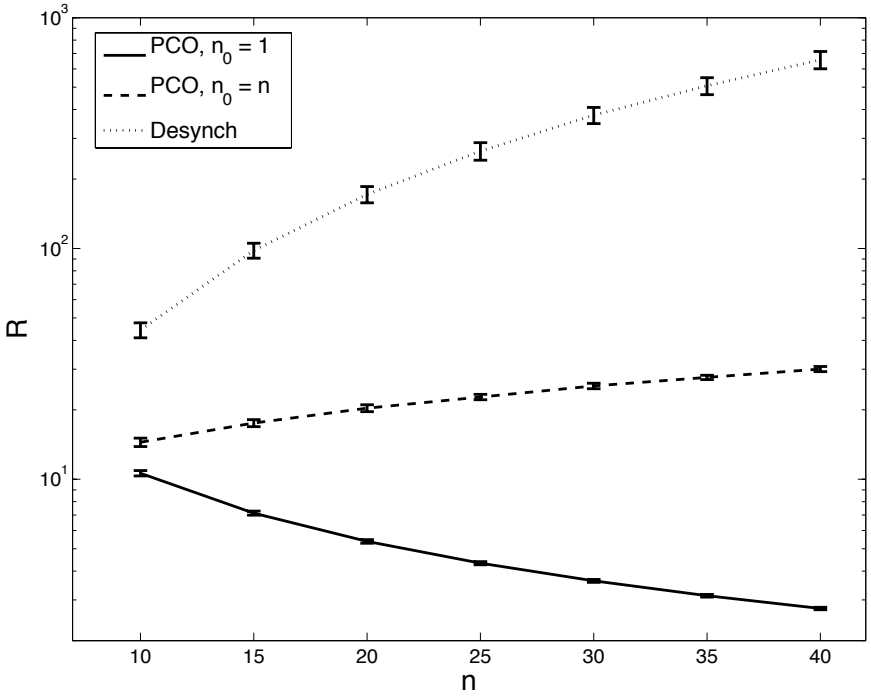

Figure 3.7: Number of rounds R required to achieve an accuracy $\epsilon = 10^{-4}$ for (i) DESYNCH with $\alpha = 0.9$, (ii) PCO with $n = n_0$ and $\alpha = 0.75$, and (iii) PCO-based with $n_0 = 1$ and $\alpha = 8 \cdot 10^{-2}$. Each point has been obtained by averaging over a set of 1000 experiments.

Proportional Fairness

In Figure 3.10, we show the performance of the proportional fair scheduling scheme based on two local clocks, as described in Section 3.7. In this simulations, we allow the nodes to join or leave the system or adjust their requests at different time instants. We can see that the proposed scheme can dynamically adapt to these changes. Specifically, suppose that the network is initially composed of 5 nodes, whose requests are $\mathbf{K} = [5, 5, 5, 20, 20]$. At round 200, nodes 4 and 5 leave the network, while nodes 1, 2 and 3 remain in the network. We can see that the period T_f is then shared equally among the three nodes, since $K_1 = K_2 = K_3 = 5$. At round 500, node 6 with demand $K_6 = 20$ joins the network. The convergent state

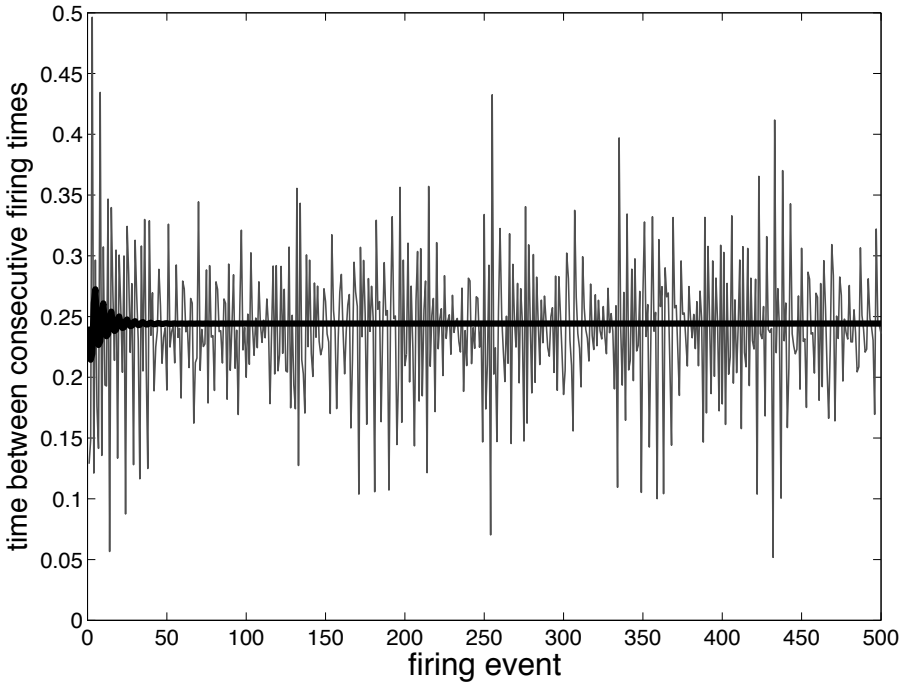

Figure 3.8: Weak desynchronization with packet losses. In black is reported the time elapsed between consecutive firings in the ideal case ($p = 0$), while in grey is reported the same quantity with $p = 0.1$. In this case $n = 10$ and $\alpha = 0.1$. We can see that in the latter case the amount of time reserved by a node is not constant.

then allots to node 6 the biggest amount of time, while the rest is allotted an equal share of the remaining time. Finally, at round 800 the requests become all equal, i.e., $K_1 = K_2 = K_3 = K_6 = 20$ and the nodes converge toward a state where they all get the same portion of time.

Proportional fairness can also be achieved using the DESYNCH protocol by having each node generate a number of virtual nodes that is equal to its demand, say K_i. Hence, the total number of virtual nodes in the network would be $\sum_{i=1}^{n} K_i$. In Figure 3.11, we compared the convergence time between DESYNCH with virtual nodes and the proposed proportional fair scheduling scheme. The parameter $K_{\max}$ has been set to 5 and 10, respectively. The request of each node has been generated

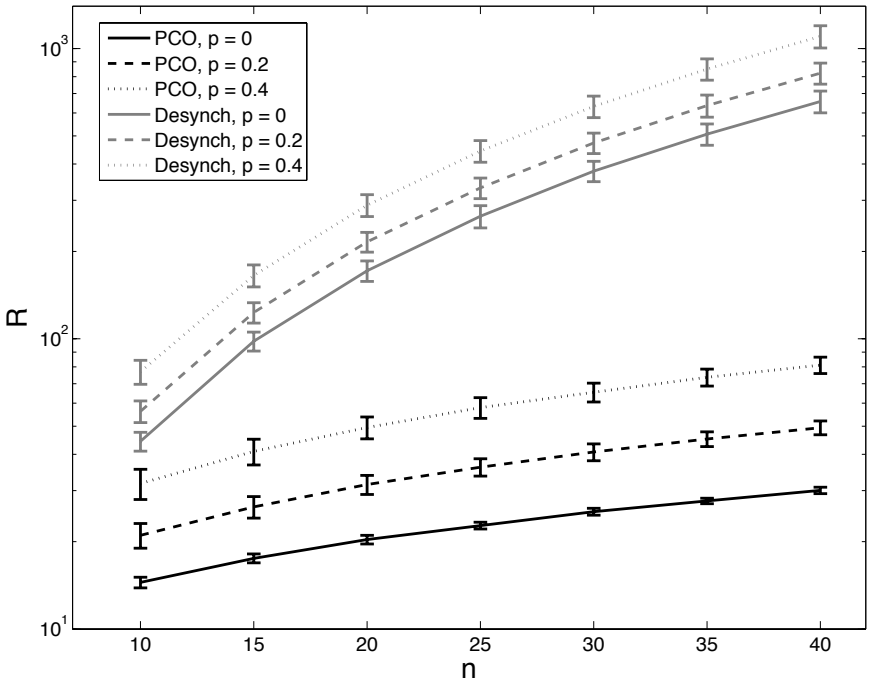

Figure 3.9: Strict desynchronization achieved by PCO and DESYNCH in the ideal case and with noisy channel. The plot shows the number of rounds needed to achieve convergence, with $\epsilon = 10^{-4}$, $\alpha = 0.75$ for PCO and $\alpha = 0.9$ for DESYNCH. Data have been averaged over 1000 experiments for each point.

uniformly at random between 1 and $K_{\max}$. We can see that the PCO-based scheme is faster than DESYNCH in both cases, even though the complexity is of the same order (which corresponds to the slope of the curve in the plot). Our scheme has the advantage that only two clocks are needed and, thus, it is easier to implement than the DESYNCH, which would require K_i clocks for each node.

In Figure 3.12, we reported the evolution of the system with the effect of detection errors. The solid line refers to the case with $p = 0$, while the dotted one shows the case with $p = 0.1$. The network is composed of 5 nodes, with $\alpha = 0.03$ and the demands are given by $\mathbf{K} = [5, 2, 3, 4, 2]$. We can see that the proposed scheme is relatively robust to detection errors. In the following Section we will

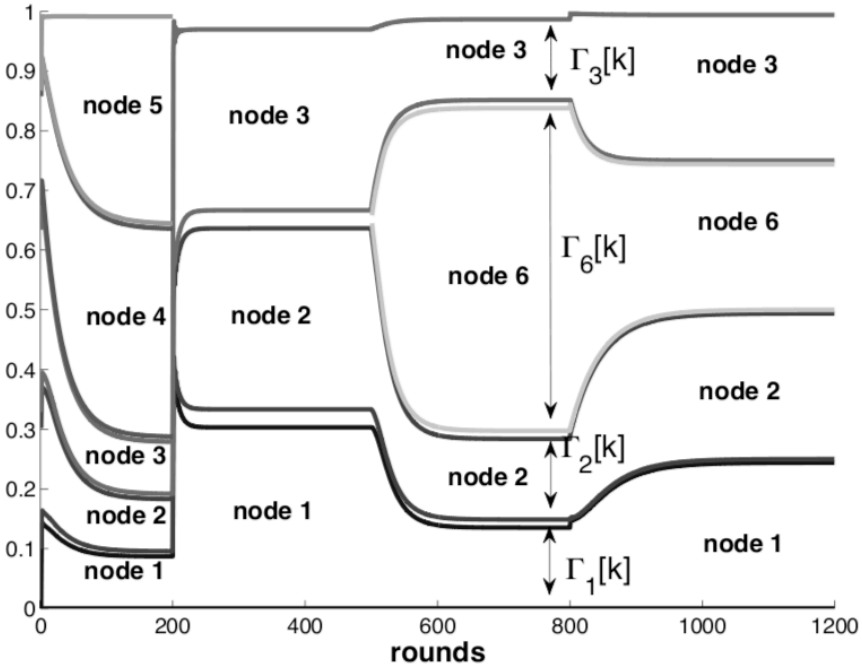

Figure 3.10: Evolution of $\Phi_{i,1}(t)$ and $\Phi_{i,2}(t)$ with respect to $\Phi_{1,2}(t)$ when nodes leave or join the network, and the requests change over the time.

study, in more detail, the implementation of our algorithm for proportional fair scheduling taking into account for non-idealities in the communication channel.

3.9 MAC Layer: Design and Implementation of PCO based MAC with UWB platforms.

Although in principle the algorithms works correctly, it is not possible in practice to measure arrival time of *pulses* with infinite precision: this is not only due to the fact that they are stored into a buffer (if digitally processed), but also to the fact that that the propagation time is non zero, and that there is noise.

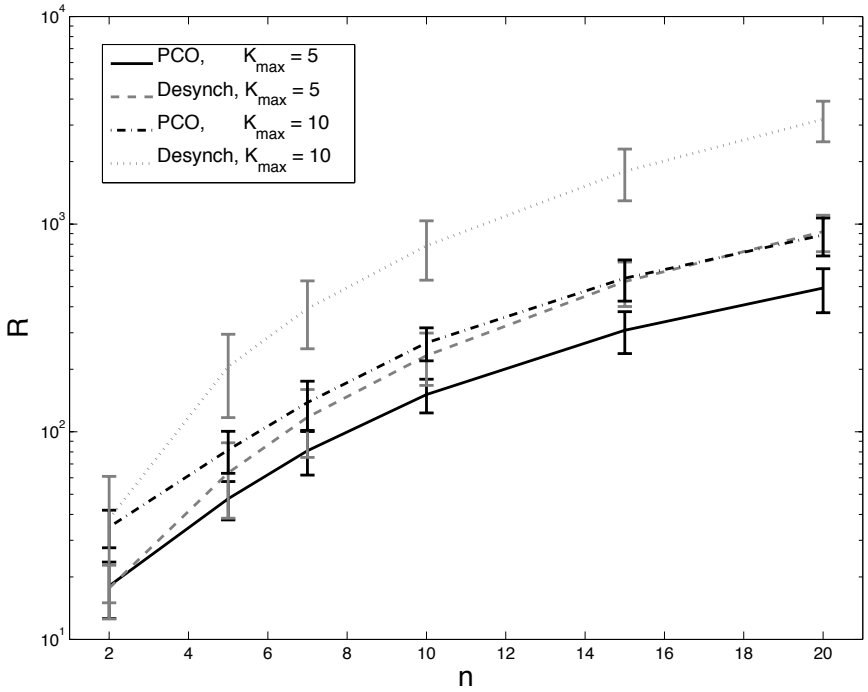

Figure 3.11: Number of rounds required to reach desynchronization for DESYNCH and our PCO-based protocol, with $\epsilon = 10^{-2}$ and $\alpha = 0.9$ in both cases. Data have been average over a set of 1000 experiments for each point.

For all these reasons, a more appropriate model should consider the time as being known within a certain accuracy and, therefore, be quantized at the corresponding precision level.

The protocol will interact with a Physical Layer design that will most likely return a time with a precision that is in the order of the symbol duration. Assuming that this is the case, in the following we indicate how to modify our proportionally fair scheduling strategy to attain the desirable effect, when the firing times are quantized.

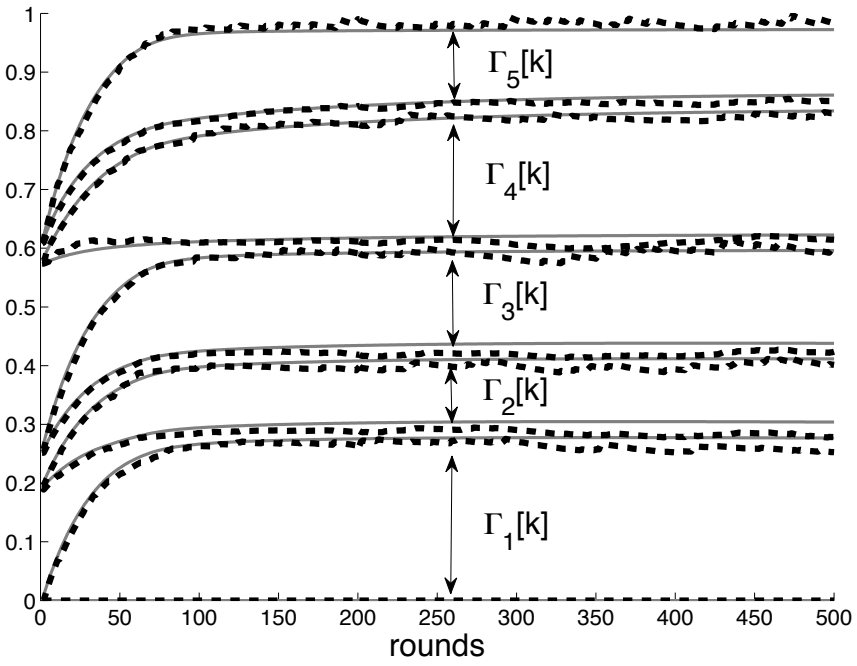

Figure 3.12: Evolution of a system composed of $n = 5$ nodes, with requests vector $\mathbf{K} = [5, 2, 3, 4, 2]$, $\alpha = 0.03$ and $p = 0.1$. In the figure are reported both the ideal case (solid line) and the case with noisy channel (dotted line).

3.9.1 Quantized Proportional Fairness

We can imagine a quantized version of the algorithm described in Section 3.7. The two clocks count time in terms of numbers of PCO frames (of duration equal to T_{PCO}), so they effectively indicate at what PCO frame node i wants to start transmitting and at what PCO frame node i has ended its transmission, as shown in Figure 3.13. Thus, the PCO hardware platform developed in [5] can be used to achieve this task. The unit step of our scheme for proportional fairness is equal to the PCO period $T_{\text{PCO}} = 7\mu$s. Consequently, the period of the algorithm is $T_{\text{PF}} = LT_{\text{PCO}}$, where L is a parameter indicating the number of frames used to form a period. Assume that at time t node i updates its two clocks using (3.14).

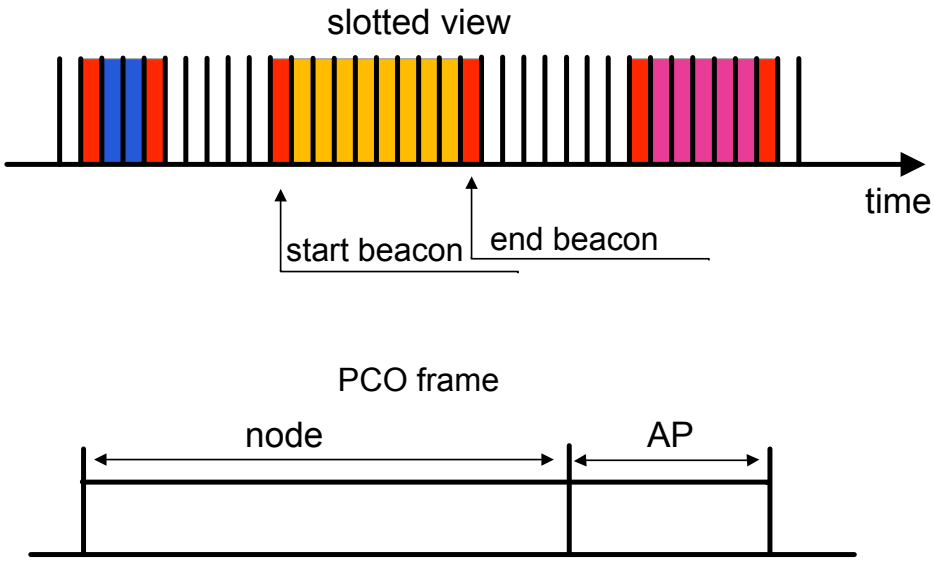

Figure 3.13: Proportional Fairness with Access Point (AP): a master node is responsible of echoing the beacons that delimit the beginning and the end of transmissions.

In order to make the state independent of the quantization noise, the new clock values will be quantized as

$$\hat{\Phi}_{i,2}(t) \triangleq \mathcal{Q}(\Phi_{i,2}(t)) = \min_j \left| jL^{-1} - (\Phi_{i,2}(t) + u) \right|$$
$$\hat{\Phi}_{i,1}(t) \triangleq \mathcal{Q}(\Phi_{i,1}(t)) = \min_j \left| jL^{-1} - (\Phi_{i,1}(t) + u) \right|$$

in analogy with the quantized consensus-type update with dithering studied in [6], where u is a uniform r.v. with support on $[-(2L)^{-1}, (2L)^{-1}]$.

3.9.2 Implementation

We propose to introduce an access point in the network, as shown in Figure 3.14. This node will utilize a fixed portion of PCO-frame period (downlink) of duration T_{PCO}, as shown in Figure 3.13, with the purpose of echoing the activities of the nodes (beginning and end of transmissions), while the rest of the period is utilized by the node currently under transmission (uplink). This mechanism makes the overall algorithm more robust to channel non-idealities letting the nodes to exploit, more safely, a new feature: *energy saving*. Each node could, in fact, keep its radio on for an amount of time sufficiently large to hear the clocks of its phase neighbors (echoed by the AP), and shut the transceiver off for the rest of the period, as we are going to discuss in the next Section.

Algorithm 5 shows the pseudo-code of a direct implementation of the PFS-PCO algorithm in an event-driven fashion, where each node utilizes the time between its own clock to transmit data to the access point. The events that need to be handled are the reception of a beacon signals and the end of the period. In the first case the node records its distance to the neighbors; in particular the beacon heard right before the local firing will be the distance to the preceding neighbor, while the first firing after the two local ones will be used to update the clock values.

3.9.3 Parameters Choice

We propose two ways of implementing the choice of K_i at the local level:

- *preference model:* each node is given a preference hardcoded in its memory, based on its priority that depends on what the sensor will be monitoring,

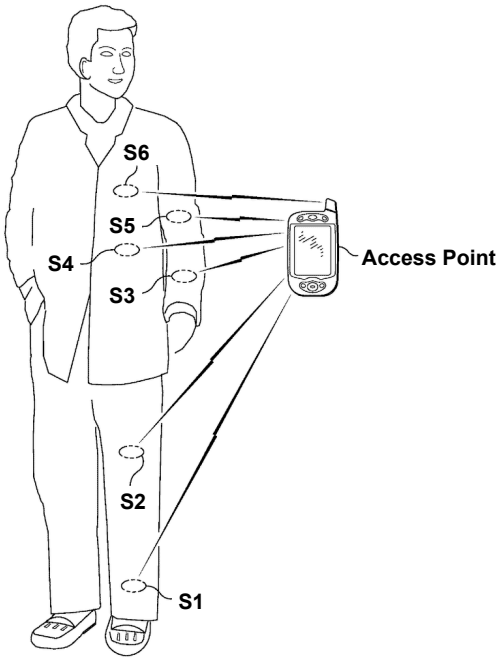

Figure 3.14: Wireless Body Area Network with Access point.

that is, if sensor i has priority over sensor j then $K_i > K_j$;

- *rate constrained model:* each node has a minimum number of bits to transmit per period, that we indicate with $d_i(t)$.[4] The current allocation, in terms of number of bits per frame assigned to node i, is indicated by $\lambda_i(t)$. If such number of bits is not at least equal to $d_i(t)$, the node increases its demands, embedded in K_i, keeping in mind that $K_i(t) \leq K_{\max}$. Therefore, in the rate constrained model, the update of parameter K_i is given by

$$K_i[m+1] = \max\left\{\lfloor K_i[m] + \mathrm{sgn}(\lambda_i[m] - d_i[m]) - K_{\max}\rfloor_+, 1\right\}$$

where $\mathrm{sgn}(\cdot)$ is the sign function and m is a multiple integer of T_{PF}.[5]

[4] In the context of WBANs, this quantity can be thought as approximatively constant.

[5] Note the analogy between (3.15) and the distributed implementation of CSMA with back-pressure [36].

Algorithm 5: Event-driven description of our proportional fairness scheme.

```
1. Receive Beacon:
2.    if( enabled )
3.       record(i+1, t); enabled = false;
4.       [start, stop] = update();
5.       set_start_clock();
6.    else
7.       record(i-1, t);
8.    endif
9. start_clock Tick:
10.   begin_TX;
11.   set_stop_clock();
12. stop_clock Tick:
13.   stop_TX; enabled = true;
```

Missed detections and false alarms introduce errors in the algorithm, since the update is based on clock firings implemented through the transmission of signals within the bins corresponding to state variables Φ_i and Ψ_i. If not properly handled, the algorithm may not be working correctly. If the coupling strength is fairly small, having an incorrect estimate of the neighbors' states does not hurt significantly the algorithm, because the node only makes small phase jumps in each iteration. However, this is not enough; we experimentally found that both probabilities (missed detection and false alarm) have to be less or equal to 0.001 in order for the algorithm to converge. In the next section we show how, by increasing the length of the PCO frame, it is possible to meet such requirement.

3.10 Impact of the Physical Layer

The transmission of clock signals corresponds to the transmission of specific beacons in the portion of PCO frame allotted to the sensors, encoding the start and end signals. In this section we describe the physical layer model of the UWB radio and show how to design the receiver for the beacons and extract information about the channel state.

3.10.1 Signaling Parameters and Model

The PCO synchronization mechanism supports the channel access technique and the subdivision of the spectrum because it allows to precisely duty cycle the radio to turn on and off the transmitter and receiver in the dedicated spots of the frame. We assume that the nodes synchronization errors are a small percentage of the bin and include the effect of asynchronism in our fading model. Let us assume that each bin has a normalized duration of one, so that in the $n = iB + b$ bin we have the b^{th} bin of the i^{th} frame. More specifically, the transmit a bandpass pulse is a waveform $g(t) = \Re\{p(t)e^{j2\pi f_0 t}\}$ where $f_0 = 4.5$GHz and where the envelope $p(t)$ is orthogonal to the PCO signal and meets the FCC mask requirements. If the channel delay τ is modest, denoting by $\sqrt{G}$ the channel amplitude gain, the reception of a on-off signal transmitted in bin n corresponds to the reception of:

$$r(t) = \Re\{\sqrt{G}x_n p(t-n-\tau)e^{j2\pi f_0(t-n-\tau)}\} + w(t) \quad (3.15)$$

$$\approx \Re\{\sqrt{G}e^{j\phi}x_n p(t-n)e^{j2\pi f_0 t}\} + w(t), \quad (3.16)$$

where $x_n = 0$ (OFF) or $x_n = 1$ (ON). As well known, the model above can also capture well the scattering that has delay spread that is negligible, i.e. whose

coherence bandwidth is larger than the bandwidth of the complex envelope $p(t)$. Start and end beacons are used to mark the data portion of each user so that no confusion between no transmission and a "zero" bit transmission is established.

Due to the low complexity requirements, the receiver is non-coherent [66]. To avoid the fading effects one should extract the absolute value square of the projection onto the signal space. Considering, without loss of generality, the signal centered in bin 0, to extract the statistics, the receiver projection consists of the projection onto the filters $p(t)\cos(2\pi f_0 t)$ and $-p(t)\sin(2\pi f_0 t)$ followed by the calculation of the absolute value square, i.e.:

$$\hat{r} = \left| \int p(t) e^{-j2\pi f_0 t} r(t) dt \right|^2 = |\sqrt{G} e^{j\phi} E_p x + w|^2, \tag{3.17}$$

where $w \sim \mathcal{CN}(0, E_p N_0/2)$. We define the decision made at the receiver as $y = \mathbf{1}(\hat{r} > \xi)$, where $\mathbf{1}(\cdot)$ is the unit-step function and ξ the decision threshold.

3.10.2 Channel Model and Threshold Choice

The wireless channel between the sensor nodes and between a sensor and the access point is assumed to follow a slow flat fading model. Hence, the random channel coefficient $\sqrt{G}e^{j\phi}$ remains fixed over one frame and varies independently from one frame to another. One typical scenario in body area networks is that the sensors are wearable by a patient and transmit to an access point located inside the patient's hospital room. In this case, the Rician distribution was shown to be a suitable model to describe the fading envelope $\sqrt{G}$ fluctuations [74]. Hence, we model $\sqrt{G}e^{j\phi}$ as $\sim \mathcal{CN}(\mu, \sigma_G^2)$. In practice, the Rician K-factor parameter will vary depending on the angle between each sensor and the access point. Using the complex Gaussian model, all is needed for the threshold choice is to derive the

likelihood function $f(\hat{r}|x)$ and what results from this assumption is that the conditional distributions $f(\hat{r}|1), f(\hat{r}|0)$ are non central and central χ-square respectively. The optimum threshold that minimizes the error probability is the one given by the Maximum A Posteriori (MAP) rule.

In practice the receiver receives from different nodes on different bins and it does not know the parameters $\mu_i, \sigma^2_{G_i}$ for the nodes so using the optimum threshold is not possible. Hence, the performance of the MAP detector is simply a theoretical bound. The question is how to set the threshold for the detection of the symbols in some reasonable way that ensures coverage of the sensors of interest. We assume that the receiver knows the noise variance at the output of the receive filter and that in our design we aim at a certain target range for the nodes the cluster head wishes to serve. For BAN this range would be around $3-6$m. So, given the average power gain $P_i = |\mu_i|^2 + \sigma^2_{G_i}$ value for a generic node within the area to cover, there will be a nominal value P^* that will be consider the average power gain at the median or average range, or at the edge of the cell. The ratio $\rho_i = |\mu_i|^2/\sigma^2_{G_i}$ is typically assumed to be $\approx \rho^*$ and a constant that is function of the type of expected deployment, since it represents the average portion of scattering component in the fading versus the line of sight component, which will not vary too much if the deployment is similar [74]. Hence P^*, ρ^* will have an associated $|\mu^*|^2 = P^*/(1+\rho^*)$ and similarly $\sigma^{*2}_G = P^*\rho^*/(1+\rho^*)$. Given the desired range and corresponding level of the signal that the receiver is optimized for, the threshold ξ is then chosen as the threshold of the MAP detector for that specific range, i.e. the threshold that is optimal for the corresponding μ^*, σ^{*2}_G.

Because of this static threshold parameter, the channel in general not only will not be a binary symmetric channel but it also will not provide the optimal

performance. More specifically, we can model the channel as a binary memoryless channel with input X and output Y where the transition probabilities are

$$\alpha_{\mathrm{b}} \triangleq P(Y=0|X=0) = \int_0^{\xi} f(\hat{r}|0)dy$$

and

$$\beta_{\mathrm{b}} \triangleq P(Y=1|X=1) = \int_{\xi}^{+\infty} f(\hat{r}|1)dy.$$

At any range α_{b} does not change, but β_{b} does, so β_{b} is a function of the transmitter receiver channel, a fact that we will highlight by using the notation $\beta_{t,i}$ to denote the effect of transmitter i. For transmitters that are further away from the receiver $\beta_{\mathrm{b},i}$ will be reduced while it will tend to one as they get very close. The probability of error of the receiver when node i transmits is given by:

$$P_{\mathrm{e}}(i) = \frac{1}{2}(1-\alpha_{\mathrm{b},i}) + \frac{1}{2}(1-\beta_{\mathrm{b},i}) = 1 - \frac{\alpha_{\mathrm{b}}}{2}\left(1 + \frac{\beta_{\mathrm{b},i}}{\alpha_{\mathrm{b}}}\right). \tag{3.18}$$

Clearly, the performance monotonically improve as $\beta_{\mathrm{b},i}$ increases.

3.10.3 The Detection of the Beacons after the On/Off Channel

To implement our Multiple Access Control (MAC) protocol, given $\{y_n\}_{n=1}^N$, we need to distinguish between the following four hypotheses: 1) H_s: the input is a start beacon $\{s_n\}_{n=1}^N$; 2) H_e: the input is an end beacon $\{e_n\}_{n=1}^N$; 3) H_d: the input are binary i.i.d. data; 4) H_0: the input is all zeros. The following Lemma summarizes the receiver identification of the beacon sequences (the proof is given in B.6).

Lemma 1 *The optimal decision for the hypothesis H_i is obtained by choosing H_i for which $\Lambda(\{y_n\}_{n=1}^N|H_i)$ is maximum, where $\Lambda(\{y_n\}_{n=1}^N|\{x_n\}_{n=1}^N)$ is equal to*

$$\sum_{n=1}^{N}\left[g(x_n,y_n)(y_n x_n \log\hat{\beta}+\bar{y}_n\bar{x}_n\log\alpha_{\mathrm{b}})+\overline{g(x_n,y_n)}(\bar{y}_n x_n\log(1-\hat{\beta})+y_n\bar{x}_n\log(1-\alpha_{\mathrm{b}}))\right],$$

for H_s, H_e, H_o, where $\overline{g(x,y)}$ is the ex-or function and

$$\hat{\beta}=\left(\left(\sum_{n=1}^{N}y_n x_n-\sum_{n=1}^{N}\bar{y}_n x_n\right)\left(\sum_{n=1}^{N}y_n x_n\right)^{-1}\right)^{+}.$$

For H_d, instead, we have that $\Lambda(\{y_n\}_{n=1}^N|H_d)=\log\frac{p}{1-p}\sum_{n=1}^{n}y_n+N\log(1-p)$, where $p=\frac{\hat{\beta}+1-\alpha_{\mathrm{b}}}{2}$ and

$$\hat{\beta}=\left(\alpha_{\mathrm{b}}-1+\frac{2\sum_{n=1}^{n}\bar{y}_n}{N}\right)^{+}.$$

3.11 Results: Part II

In Figure 3.15 we report an example with $n = 5$ nodes, with requests vector $\mathbf{K} = [5, 5, 1, 1, 5]$ and 256 frames per period. The horizontal axis is the iteration index, while the vertical axis represent the amount of period obtained by the nodes. In this case, the coupling parameter is $\alpha = 0.03$ (small jumps), $K_{\max} = 5$ and $\delta = 2$. The probabilities of missed detection and false alarms are $P_m = 10^{-3}$ and $P_f = 10^{-4}$. We can see that the state of the network is quickly established, and the nodes obtain an amount of time proportional to their demands. Occasionally, some spikes occur because either missed detections or false alarms cause the nodes to get incorrect estimates of their neighbors state. The empty spots are about 0.08 (21 frames) as forecasted by the theoretical results. In the same figure is also reported the evolution of the system under idealistic conditions (in grey), i.e., with no quantization and no errors due to channel noise.

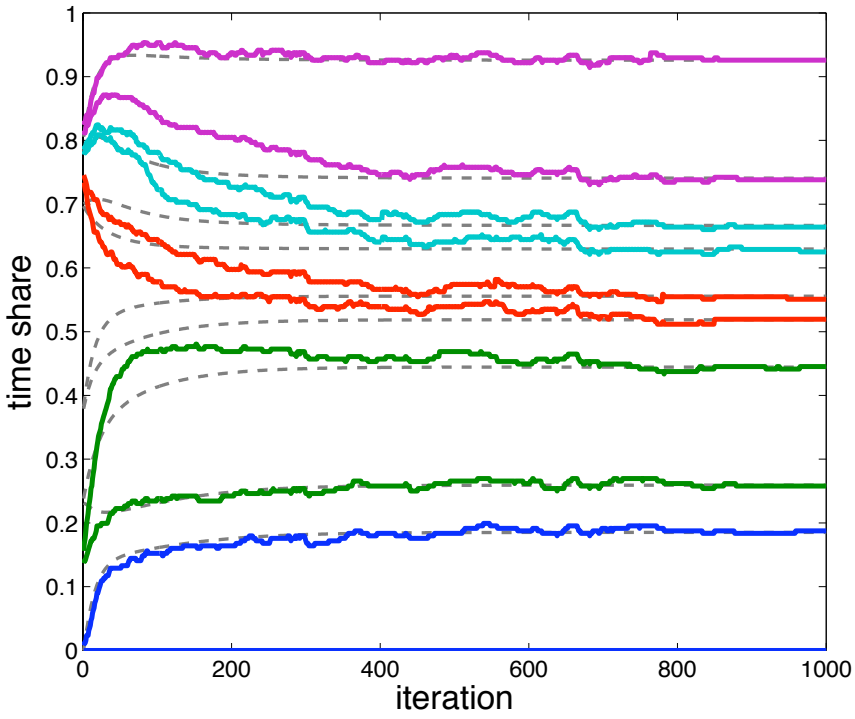

Figure 3.15: Proportional Fairness: the space between two lines of the same colors is the amount of time obtained by a node (from node 1 to node 5, bottom-up). The request vector is $\mathbf{K} = [5, 5, 1, 1, 5]$, $\alpha = 0.03$ and $\delta = 2$. In grey is reported the evolution of the system under ideal conditions.

In Figure 3.16 we allow the nodes to leave and join the network. Within the first 500 iterations, 5 nodes are present in the network with $\mathbf{K} = [4, 4, 1, 1, 1]$. At iteration 500 a new node, node 6 joins the network with demand $K_6 = 1$. At iteration 1000, 2 nodes leave the network (nodes 3 and 6) and the request vector becomes $\mathbf{K} = [4, 4, 1, 1]$. We can see that, in all cases, the network reaches a new equilibrium corresponding to the ratios of the nodes' demands.

In Figure 3.17 are reported the probabilities of false alarm and missed detection based on the detection rule we discussed above, as a function of the beacon length. In order to include the effects of noise and fading we set $\alpha_{\mathrm{b}} = 0.9$, while $\beta_{\mathrm{b},i}$ is uniformly distributed in $[0.75, 0.9]$ and estimated using the GLRT. The start

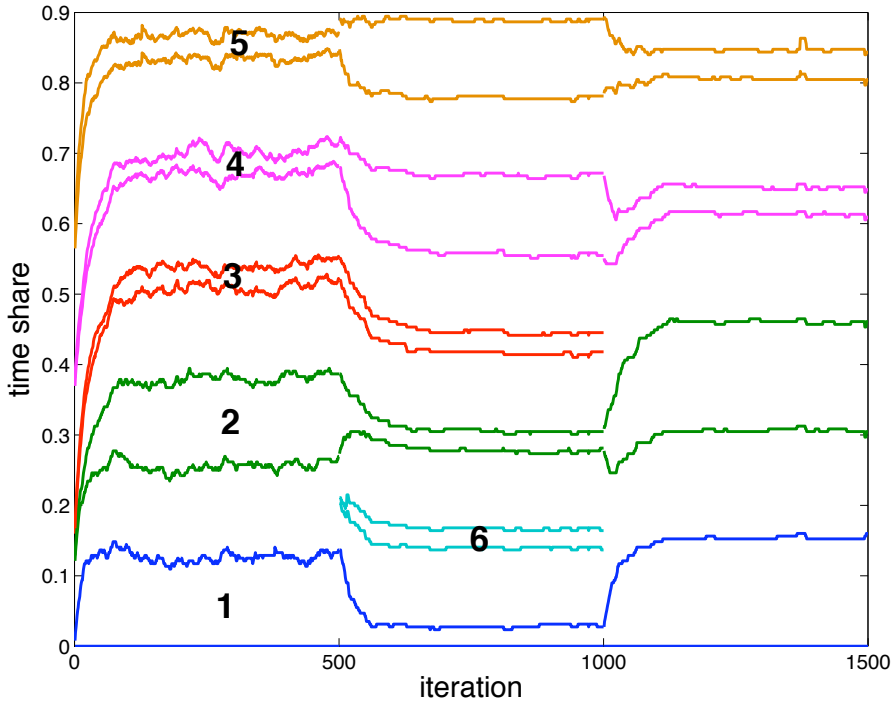

Figure 3.16: Proportional Fairness when nodes are allowed to join and leave the networks. Here, $\alpha = 0.05$; the number of frames per period is 256 and $P_m = P_f = 10^{-3}$. The amount of space between lines of the same color corresponds to the amount of time obtained by the node.

beacon codeword consists of the first half bits equal to one and the remaining equal to zero (and its complement is used as end beacon). We can see that for reasonable values of the beacon length, such probabilities are less than 10^{-3}. There is a dependency between probabilities of missed detection and false alarm on one hand, and coupling strength, demands and empty spots on the other hand. Increasing the coupling strength, causes the nodes to adjust their clocks more aggressively and, thus, false alarms might cause overlapping. This can be mitigated by using bigger empty spots (i.e., by increasing δ), and using a small value of $K_{\max}$, typically less than 10.

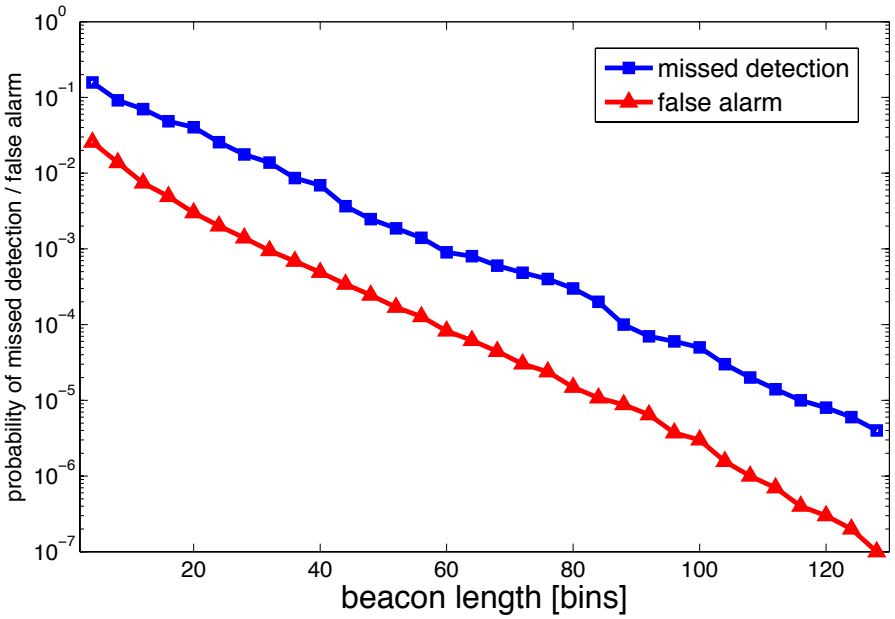

Figure 3.17: Probability of missed detection and false alarm. $\alpha_{\mathrm{b}} = 0.9$ and $\beta_{\mathrm{b},i}$ is uniformly distributed in $[0.75, 0.9]$, and estimated using the GLRT.

Although the protocol we proposed is inherently simple from an algorithmic perspective, we can see that many trade-offs are involved in choosing the right values for all these parameters. An accurate study of such dependencies goes, however, beyond the scope of this Thesis. Future work will be dedicated to the implementation of such primitive for proportional fair schedule for sensor platforms.

CHAPTER 4

NONLINEAR VOTING MODELS AND THEIR APPLICATIONS

4.1 Motivation and Related Work

The problem of coordinating a group of mobile nodes moving together in a controlled environment has been widely studied. At the physical layer, the wireless medium is not reliable, in part because of the limited power at which the nodes can transmit their messages, in part because it is shared and interference prone, and in part because of the stochastic nature of the communication channel itself. The design of lightweight and robust protocols that do not require much information about the network topology, yet, providing a certain degree to robustness to non-idealities in the communication channel, represents a fundamental challenge from an engineering point of view.

Recently, various forms of average consensus [13, 24, 35, 64, 87], as well as nonlinear network dynamics generated by the Kuramoto model have been applied to solve formation control problems [17, 42, 43, 45, 58]. In all cases, each node is modeled as an independent entity, whose state variable depends upon its local value and its neighbors' states.

In all the aforementioned models, the network evolves step by step, eventually converging to a state where all the nodes' variables are equal and, as in the case of consensus, equal to the average value of the initial states. If, for instance, these variables are directions, as the protocol advances the nodes will be progressively heading toward the same direction. What is proposed in literature has drawbacks that our method addresses. In synchronous average consensus, the updating rule

requires perfect scheduling and a coupling parameter that is a function of the maximum degree of the network (which may be, in general, unknown) [87]. The Kuramoto model can achieve convergence in complete networks (in time), where each node always has a chance to reach any other node in the network through multiple hops. However, when this assumption is not satisfied, consensus is not always seen, unless the coupling strength is of the order of the network size n, and $n \to \infty$ [3].

Asynchronous average consensus [12, 23, 87] and some variants, such as [7], do not require network state knowledge and work in multi-hop scenarios as well as switching topologies. However, the realization of the connectivity graph supporting the state information exchange as well as the updates' timing and structure are directly dependent on the underlying communication network. In other words, information exchange among nodes, in many applications, is severely impacted by the design and limitations of the communication schemes. For instance, the rate of information update, the precision of state updates, and the communication energy are shown to directly determine the cost and the stability of the coordinated movement [42]. Authors in [39,65,88] investigated how the finite rate of the digital links affected the average consensus algorithm, by considering the *quantized consensus problem*. One important drawback of quantized consensus is the existence of a *consensus distribution* in which nodes' state updates preserve the average but do not agree [39]. In [6] it has been shown that introducing probabilistic quantized (dithered) information, the expected value of the consensus is equal to the average of the original sensor data. Quantized consensus, however, does not address explicitly the issue of interference.

In this work, we investigate the possibility of coordinating a group of mobile

agents so that their state, for example, their headings, will converge toward a common value, by means of local negotiations among the nodes that leads to a state update which is akin to a vote. In such a setting, reaching a common state is more important than preserving the average.

Unlike traditional consensus protocols, we propose a novel method based on a stochastic updating rule performed locally by the nodes. The state of each node is described by a state machine, whose evolution depends on the interactions between the node itself and its neighborhood. The whole system can then be viewed as a Markov chain, composed of n state machines whose interactions depend on the network topology. By construction, the absorbing state of this Markov chain represents the condition of agreement, *i.e.* a state where the nodes agree on a common value, and is eventually reached by negotiating over time.

The advantage of the class of algorithms proposed is that they do not require topology information nor the knowledge of the degree distribution, they are completely asynchronous, and we show that they are amenable to a simple cross-layer implementation, which exploits the nature of interference in the medium. Because of these reasons, our scheme is robust to node failures and switching topologies and provides a solution that is suitable for harsh media, such as underwater acoustic communications, where synchronization errors and network congestion cannot reasonably be removed from system models. Our protocol eliminates the problem of interference in a way that is reminiscent to that in [41]. In contrast to [41], however, our protocol does not require network synchronization.

4.2 Problem Statement and Assumptions

We consider a network composed of n nodes, indicated by $\{1, 2, \ldots, n\}$. Each node has a local state variable $\theta_i(t)$ that indicates its heading[1]. We address the following two problems:

Problem 1: Is it possible for the nodes to reach, by local interactions, a condition where $\theta_i(t) = \theta_j(t), \forall i \neq j$?

Problem 2: If so, could this scheme be employed in a scenario where a leader wants to coordinate a set of other nodes (followers) via local interactions?

In literature, the first issue is referred to as *alignment*. The second one is referred to as *leader-follower*. To address the rate constraint, we assume that each value of $\theta_i \in [0, 2\pi)$ is quantized according to a predefined quantization scheme $\mathcal{Q}$ to a value $\hat{\theta}_i = \mathcal{Q}(\theta_i)$. This value is then mapped through a one-to-one function $f(\cdot)$ to an auxiliary state variable c_i that we call color, i.e., $f : \hat{\theta}_i \rightarrow c_i$. For example, if $\mathcal{C}$ is the set of available colors (and $C = |\mathcal{C}|$ its cardinality), in the case of uniform quantization:

$$c_i = \underset{k}{\operatorname{argmin}} \left| \theta_i - \frac{2\pi k}{C} \right|, k \in \{0, \ldots, C-1\},$$

and, hence $c_i = f(\hat{\theta}_i) = \frac{C}{2\pi}\hat{\theta}_i$. We assume that the following conditions hold.

A.1: Each state variable $c_i \in \mathbb{Z}^+$ belongs to finite set of elements $\mathcal{C} = \{0, \ldots, C-1\}$. This set is known *a-priori* to each node.

A.2: The updates occur at random times. The wake-up times are described by a Poisson process, and the probability that more than one node performs the

[1]The same reasoning would apply if $\theta_i(t)$ were a velocity or a combination of both the heading and the velocity.

update at the same time is zero.[2]

A.3: A node performs an update right before transmitting, based on information it has accumulated since its previous transmission terminated. The specific way information is encoded and processed is explained in the next sections.

4.3 Algorithm and Convergence Properties

We use an index k to identify the sequence of updating events in the network and to track the evolution of the states with respect to this index. That is, $k = 1$ corresponds to the first update, $k = 2$ to the second, and so on. Corresponding to each of these events, we define the variable $c_i(k)$ as the state of node i after the k^{th} update, while the variable $\mathcal{I}(k)$ indicates the node performing the update at iteration k.[3] We associate to each node a set $\mathcal{U}_i = \{k : \mathcal{I}(k) = i\}$ which indicates the set of events corresponding to the updates of node i. If $u_i^{(s)}$ is the s^{th} element of $\mathcal{U}_i$, the time elapsed between $u_i^{(s-1)}$ and $u_i^{(s)}$ is the time elapsed between the $(s-1)^{th}$ and the s^{th} updates of node i.

Algorithm 1: a generic node i records the colors received between two consecutive updates $u_i^{(s)}$ and $u_i^{(s+1)}$ and stores them in a histogram, so that for each color $l = 0, \ldots, C-1$ the quantity

$$Q_{il}^{(s)} = \sum_{k=u_i^{(s-1)}+1}^{u_i^{(s)}-1} \delta[c_{\mathcal{I}(k)}(k) - l] \tag{4.1}$$

[2]Similarly to [13], if τ is the propagation plus processing time, the probability that two updating events fall within an interval of time equal to τ (concurrent updates) is negligible as long as $\lambda\tau << 1$. So, given τ, λ can be chosen appropriately.

[3]For example, $\mathcal{I}(2) = 1$ indicates that the second updating event in the network is performed by node 1.

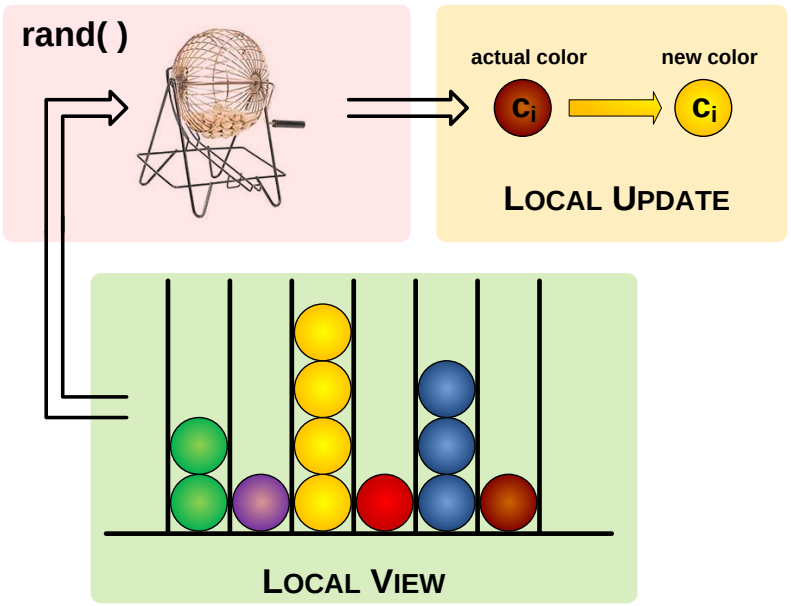

Figure 4.1: Key idea behind our gossip-based algorithm. The figure illustrates the update performed by a generic node of the network, based on the colors received by its neighborhood.

indicates the number of times color l has been reported to node i.[4] At time $t(u_i^{(s+1)})$ node i performs the update of its state variable (i.e., its color $c_i(u_i^{(s+1)})$), broadcasts the new value to the neighborhood and resets its histogram. The new state $c_i(u_i^{(s+1)})$ is chosen at random, according to the following probability density function:

$$c_i(u_i^{(s+1)}) \sim \mathcal{P}_i(x, u_i^{(s)}) = \sum_{l=0}^{C-1} q_{il}(u_i^{(s)})\delta(x-l) \tag{4.2}$$

where $q_{il}(u_i^{(s)}) = \frac{Q_{il}(u_i^{(s)})}{\sum_{m=0}^{C-1} Q_{im}(u_i^{(s)})}$ and, thus, $\sum_{l=0}^{C-1} q_{il}(u_i^{(s)}) = 1$. In Section 4.7, we further explain how the colors are exchanged and how the histograms is formed. In Figure 4.1 is illustrated the key idea behind this algorithm.

[4]Notice that the number of colors used by node i to perform the update is not necessarily equal to the number of its neighbors. Moreover, by definition of $Q_{il}^{(s)}$, within the interval $(u_i^{(s-1)}, u_i^{(s)})$ we have that $c_{\mathcal{I}(k)} \neq i$.

4.4 System Model based on Automata

The algorithm above is radically different from typical consensus or Kuramoto-based schemes [3, 6, 12, 13, 17, 24, 35, 39, 42, 58, 64, 65, 87]. While in those cases the updating function is deterministic, in our model the update is stochastic, due to the choice of the new state given by Eq. (4.2). The state update can be viewed as a node casting a vote at random, where the distribution of the vote is biased by the votes expressed by the neighbors that were heard since the last vote.

The state of each node can be described by a $(C+1)$-dimensional vector. The first element is the color of the node, while the remaining C elements, that we call *buffer*, are the number of times each of the C available colors has been received from the last local updating event up to that point.[5]

Since the the updating sequence is random, and the evolution only depends on the most recent update, the entire system can be viewed as a discrete Markov chain. The state of the network is then given by a $(C+1)n$-dimensional vector, given by the union of the nodes' states. Clearly, the condition where all nodes have the same color, and no other colors appear in any of their buffers, is an absorbing set, since a node cannot choose a color that does not appear in the network. *The idea is that, as the algorithm proceeds, the number of colors in the network eventually decreases, and therefore, nodes are naturally driven towards the same color which represents the state of agreement.*

[5]For example, suppose that there are 3 available colors, and that at iteration k node i has received, from its last update, colors 0 once and 2 twice, and that $c_i(k) = 1$. Then, the state of the node at iteration k is given by the vector $(1, 1, 0, 2)$. If node i updates at iteration $k+1$, its new color will be chosen according to the probability mass function induced by the histogram $(1, 0, 2)$. If the new color is, for instance, $c_i(k+1) = 2$, then the state of node i becomes $(2, 0, 0, 0)$. On the other hand, all nodes receiving the new state of i will increment their count value for color 2 by one.

Theorem 6 *Consider a connected network composed of n nodes performing Algorithm 1. For any iteration $k > 0$, condition*

$$\Pr\left(c_i[k+n] = c_j[k+n],\ \forall i, j \in \{1, 2, \ldots, n\}\right) > 0$$

holds true.

We can now use the result provided by Lemma 6 to establish the following convergence result.

Corollary 1 (Alignment with no Leader) *Consider a connected network composed of n nodes performing Algorithm 1, with initial conditions $\{c_1(0), c_2(0), \ldots, c_n(0)\}$ and $c_i(0) \in \{0, 1, \ldots, C-1\}, \forall i$. The event*

$$\exists\ k, c \in [0, C) :\ \forall k' > k,\ c_i(k') = c\ \forall i$$

occurs almost surely.

Algorithm 2: the algorithm we described causes the nodes to converge to a color $c \in \{0, 1, \ldots, C-1\}$. However, no guarantee exists that the final coloring of any node will be $c_1(0)$ rather than $c_2(0)$ or any other in the initial set of colors. Here, we consider Problem 2 mentioned above, where one single node is elected (or set) as a leader, while the others are the followers. In this scenario, the objective of the network is to align the followers in the same direction (color) of the leader (w.l.o.g. assumed to be node 1). To accomplish this task, we need to slightly modify Algorithm 1 as follows: the updating rule of any follower ($i > 1$ in this case) is the same as in Algorithm 1. The leader (node 1), instead, will keep its own color ($c_1(0)$) and never change it. The convergence of this scheme is established by the following result.

Corollary 2 (Alignment to the Leader) *Consider a connected network composed of n nodes performing Algorithm 2, with initial conditions* $\{c_1(0), c_2(0), \ldots, c_n(0)\}$, *and* $c_i(0) \in [0, C), \forall i$. *The event*

$$\exists\ k:\ \forall k' > k,\ c_i(k') = c_1(0)\ \forall i$$

occurs almost surely.

Discussion: both Algorithms 1 and 2 exhibit some nice features such as, for example, the fact that a node does not need to know the topology of the network, nor its local degree or the degree of its neighbors. Moreover, switching topologies are naturally handled, since the system is always characterized by the same discrete Markov chain, and, thus, the convergence properties are preserved. In addition, in case of link failures (message drops), both algorithms converge. It is worth mentioning that if the leader disappears, in the case of Algorithm 2, consensus will still be reached, in contrast to other leader-follow algorithms which outright fail if the leader disappears. Finally, we would like to emphasize that the simplicity of the updating rule in Eq. (4.2) is appealing for sensor platforms constrained in terms of processing and communication capabilities.

4.5 Nodes' Dynamics

Unlike other consensus algorithms, for any mapping from colors to states, no guarantee exists that the new state will be close to the previous one. In other words, if the state of the system $\hat{\theta}_i$, which represents the heading of node i, were to follow directly the color changes at node i, then a node would be forced to change its heading abruptly and frequently (before reaching convergence). However, having

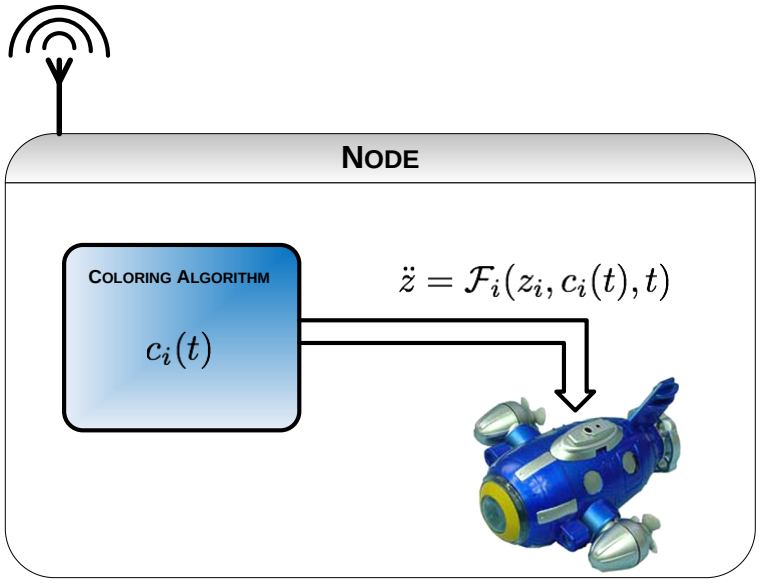

Figure 4.2: Functional representation of each node: one block (left) is responsible for the coloring algorithm, while the other handles the mechanical hardware.

fast variations in the physical movement of the nodes is undesirable. To overcome this drawback, we separate the functional node structure into two parts: one component dedicated to coloring, and another responsible for the mechanical hardware (see Figure 4.2).

For the mechanical hardware component of the nodes, introduce the state variable z_i for vehicle heading. This state variable is not exchanged among the nodes, and the changes in color are treated as a control input to the dynamics of this state, so that we can assume that z_i is a continuous function of time. Now, consider the s-th update of node i at time $t(u_i^{(s)})$. Right before the update, the color of node i is $c(u_i^{(s-1)})$, while the new color is $c(u_i^{(s)})$, and this color will not change until the next update, at time $t(u_i^{(s+1)})$. The heading of node i within the interval

$[t(u_i^{(s)}), t(u_i^{(s+1)})]$ is then controlled according to the dynamics

$$\dot{z}_i = \mathcal{F}_i(z_i, c_i(t), t) \tag{4.3}$$

where $\mathcal{F}_i$ is a user-defined function operating as a controller of the mechanical dynamics, and the dynamics are subject to the boundary condition $z_i(c(u_i^{(s-1)}), t(u_i^{(s)})^-) = z_i(c(u_i^{(s)}), t(u_i^{(s)}))$. With this construction, the state z_i will be continuous, but potentially will have non-smooth transitions between color values. To smooth these transitions, one can further extend the mechanical dynamics to the form

$$\ddot{z} = \mathcal{F}_i(z_i, c_i(t), t) \tag{4.4}$$

with the additional boundary condition $\dot{z}_i(c(u_i^{(s-1)}), t(u_i^{(s)})^-) = \dot{z}_i(c(u_i^{(s)}), t(u_i^{(s)}))$. The objective of designing $\mathcal{F}_i$ is to provide asymptotic convergence of z_i to $\hat{\theta}_i \triangleq f^{-1}(c_i(t(u_i^{(s)}))$.

In the simplest case, the dynamics of the relation between the mechanical state and color will be linear with the dynamics acting to smoothen the changes in color value. The controller can then be chosen via standard feedback methods to create an asymptotically stable tracking of $z_i(t)$ to the reference value of $\hat{\theta}_i$:

$$\ddot{z}_i(t) = -\alpha_i(z_i(t) - \hat{\theta}_i) - \beta_i \dot{z}_i(t) \tag{4.5}$$

where $\lim_{t\to\infty} z_i(t) = \hat{\theta}_i$ as long as $\alpha_i, \beta_i > 0$. More generically, similar equations and controllers can capture nonlinear dynamics and higher dimensional state dynamics (heading control incorporated into systems with both heading and position dynamics).

4.6 Scaling Laws for Different Graphs

We model the communication network as a connected random geometric graph $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ the set of edges. The neighborhood of node i is the set of nodes whose distance is less than the transmission radius r. The adjacency matrix $\mathbf{A}$ induced by graph G is an n-by-n matrix such that $A_{ij} = 1$ if nodes i and j, with $i \neq j$, are connected, and 0 otherwise. The degree matrix $\mathbf{D}$ is a diagonal matrix whose i^{th} element on the diagonal D_{ii} is the degree of node i (the number of neighbors of node i).

In the original model presented in Section 4.3, wake-up times are modeled as a Poisson process, so there is no guarantee that a node will receive exactly one message for each of its neighbors. However, for mathematical tractability, we assume that at each iteration a node, chosen uniformly at random in the network, wakes up and performs the update based on the current state of its neighbors.[6] This is equivalent to saying that a node wakes up at iteration k and copies the value of one of its neighbors, chosen uniformly at random.

We consider Algorithm 1, the case where all nodes align on the same color without leader and assume that there are n different colors, one for each node.[7] As we mentioned, the update is akin to a vote cast by each node, whose distribution is biased by the votes of the neighbors. Recognizing the connection with a *voting model*, the analysis of this algorithm can be done by following the so called *duality approach*, that maps the voting dynamics into a *coalescing random walk* on graph G. This model has been employed in the analysis of interacting particle systems

[6] This is an approximation of the original algorithm, and, as we will see in the numerical section, the theoretical results we present are confirmed.

[7] As we will see in the numerical section, the result we obtain also holds in the case of Algorithm 2.

(see e.g., [47]). The basic idea is as follows: there is a particle for each vertex at time $t = 0$. Each of these particles performs an independent continuous random walk on the graph, i.e., each particle jumps to one of its current neighbors independently according to a rate 1 Poisson process. When two or more particles meet on a vertex, they coalesce and form one single particle. Thereafter, these particles walk together on the graph G, possibly colliding with others. The *voter model* reaches consensus (all nodes have the same color) when in its corresponding dual process all particles collapse into a single one. The following result is an upper-bound of the expected convergence time $E[T_{\text{crw}}]$ for a coalescing random walk process on a given graph.

Lemma 2 (Aldous [4]) *Assume $E_i[T_j]$ is the mean hitting time of a regular random walk to node j on graph G, given that the walk is initialized at node i. Then*

$$E[T_{\text{crw}}] \leq e \log(n+2) \max_{i,j} E_i[T_j].$$

The mean hitting time is given by the following lemma.

Lemma 3 (Lovasz [49]) *The mean hitting time of a regular random walk to node j on graph G, when the walk is initialized at node i, is equal to*

$$E_i[T_j] = 2|\mathcal{E}| \sum_{k=2}^{N} \frac{1}{1 - \lambda_k(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})} \left(\frac{v_{kj}^2}{D_{jj}} - \frac{v_{kj}v_{ki}}{\sqrt{D_{jj}D_{ii}}} \right)$$

where v_k is the eigenvector of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ corresponding to eigenvalue λ_k.

Hence, based on Lemmas 2 and 3, we are able to bound the convergence time of the *voter model*.

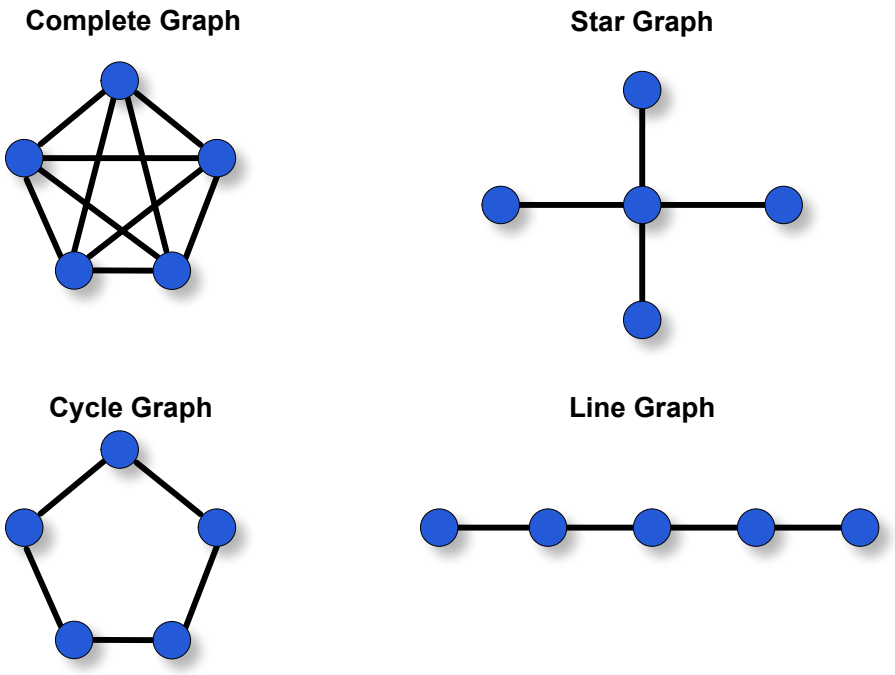

Figure 4.3: The deterministic topologies we consider are 1) the complete graph, 2) the star graph, 3) the cycle graph and 4) the line graph.

Theorem 7 *Given a network $G(\mathcal{V}, \mathcal{E})$, its adjacency and degree matrices* $\mathbf{A}$ *and* $\mathbf{D}$ *respectively, the expected time for the voter model to convergence is upper bounded by:*

$$E[T] \leq \frac{4e \log(N+2)|\mathcal{E}|}{1 - \lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})} \max_j D_{jj}^{-1}. \tag{4.6}$$

where λ_2 is the second largest eigenvalue of matrix $\mathbf{M} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.

4.6.1 Deterministic Network Topologies

In the context of deterministic network topologies we consider five types of graph, namely, 1) the complete graph, 2) the star graph, 3) the cycle graph and 4) the line graph, sketched in Figure 4.3. Consider, for example, the complete graph. In

this case, the number of edges is equal to $|\mathcal{E}| = n(n-1)/2$. The degree is equal to $n-1$ for all nodes, and $1 - \lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = n/(n-1)$. By using the result in Theorem 8, we obtain

$$E[T] \leq e(n-1)\log(n+2).$$

In a similar fashion one can compute the bound on the time to convergence for the other cases, that are reported in Table 4.1.

4.6.2 Random Geometric Graph

Let us consider the case of 2-dimensional random geometric graphs (which is the case of interest), where n nodes are randomly placed (w.l.o.g.) in a 1-by-1 square, and two nodes communicate if their distance is less than r. The following result holds true (for proof, see Appendix C.5).

Theorem 8 *Given a 2-dimensional random geometric graph with* $r \geq \sqrt{\frac{4\log n}{n}}$, *there exists a finite* $n' \geq 0$ *such that for all* $n \geq n'$, *the expected time to reach consensus is upper-bounded by*

$$E[T] \leq \frac{2e\log(n+2)n}{r^2}. \tag{4.7}$$

Table 4.1: Time to Convergence

Graph	Bounds on Expected Time to Convergence
Complete Graph	$e(n-1)\log(n+2)$
Star Graph	$e(2N-2)\log(n+2)$
Cycle Graph	$en^2\log(n+2)$
Line Graph	$e(n-1)^2\log(n+2)$

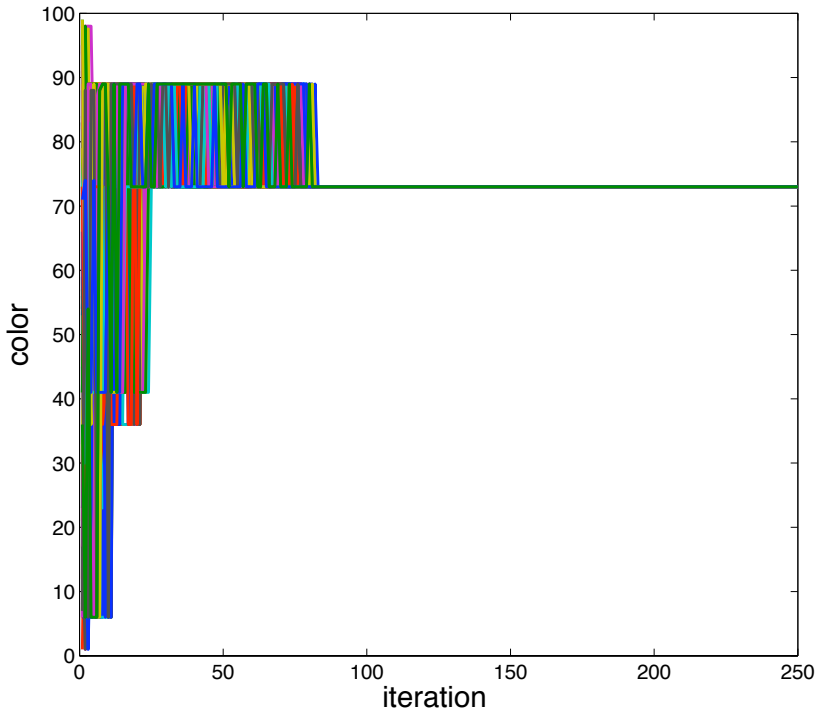

Figure 4.4: Voting algorithm performed on a network composed of $n = 100$ nodes, with normalized transmission radius $r = 0.2$.

In order to see how the performances of this algorithm scale as the network grows, we may set the transmission radius to $r = 2\sqrt{\frac{\log n}{n}}$; that is, as the network becomes large, the connectivity remains constant [14]. By substitution, from (4.7) we obtain

$$E[T] \leq \frac{e}{2} n^2 = O(n^2).$$

In Figure 4.4 we report, as an example, the evolution of the network state (the nodes' colors) over time, with a network composed of $n = 100$ nodes and normalized communication radius $r = 0.2$.

4.6.3 Erdos-Renyi Graphs

The Erdos-Renyi graph (also known as ER graph) is one of the most famous, and still used, random graph used to model social networks, because of its simplicity of analysis due to the independence of the links between the nodes. The construction of an ER graph is simple: pick a set of n nodes with no edges. For each pair of nodes, out of $n(n-1)/2$ possible combinations, establish a link with probability p (coin flip). In this context, we wish to derive a bound of the time to convergence for the voter model when the network grows, but the connectivity remains constant. To accomplish this task we introduce a dependency of p on the network size n. In order to bound the convergence time for ER graphs, we first introduce the following result.

Lemma 4 *Assume random variables $X_1, X_2, \ldots, X_N$ are i.i.d with $X_i \in \{0, 1\}$. Then,*

$$P\left(\sum_{i=1}^{n} X_i > (1+\delta)nE[X_i]\right) < e^{\frac{nE[X_i]\delta^2}{3}}, \tag{4.8}$$

$$P\left(\sum_{i=1}^{n} X_i < (1-\delta)nE[X_i]\right) < e^{\frac{nE[X_i]\delta^2}{2}}, \tag{4.9}$$

where $0 < \delta < 1$.

Of note is that the above lemma gives exponentially decaying bounds on the tail distributions of sums of independent binary random variables.

For a given node i, the number of neighbors of that particular node is equal to:

$$d_i = \sum_{j=1}^{n} A_{ij}.$$

We note that above definition introduces self-loops, *i.e.*, each node is neighbor to itself. Such an assumption is for the mathematical brevity, and does not change

the validity of our analysis. We note that due to the independent nature of the link generations, A_{ij}'s are i.i.d. binomial with mean p for a given i.

We can now apply the result of Theorem 8 to bound the expected time to consensus for ER graphs (for proof, see Appendix C.6).

Theorem 9 *Given an Erdos-Renyi graph with* $p >> \log^2(n)/n$*, the expected time to reach consensus scales as:*

$$E[T] \sim O\left(\frac{4e\log(n+2)n\ (1+2\sqrt{\log(n)/np})}{1-\frac{8}{np}-\frac{g(n)\log^2(n)}{np}}\right),$$

where $g(n)$ *is a function tending to infinity arbitrary slowly.*

The number of colors determines the achievable accuracy in the final heading of the nodes, which is equal to $2\pi C^{-1}$ radians. Rather than mapping each color directly into one direction, one can use a code to indicate, via successive levels of refinement, a specific region in the 2-dimensional circle. Suppose the number of colors is, for example, equal to 2. The first element of the code can be used to indicate whether the heading of a node is within the two regions $\{[0,\pi),[\pi,2\pi)\}$. Another 2 letters in the code can be used to further divide each of those regions into two smaller regions, and so on and so forth, as shown in Figure 4.5. So, instead of running the algorithm with C available colors, we can run multiple instances of the algorithm, corresponding to the number of levels we need to achieve a required level of accuracy. Each message is then encoded into a codeword of colors whose entries are updated independently. We will refer to each partition as a different *zooming level*.

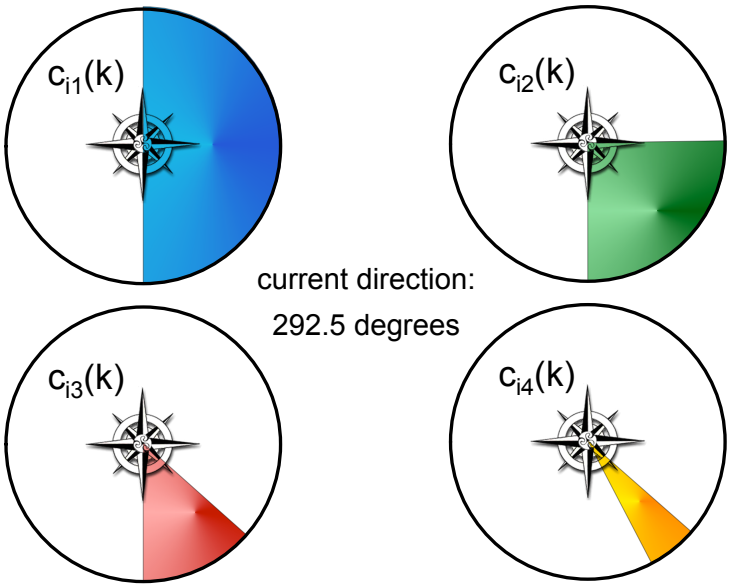

Figure 4.5: Consensus algorithm with partitioning: in this case four algorithms run in parallel with two available colors. Each instance of the algorithm determines the region in which the heading is directed in a different zooming level.

Given that P instances of the algorithm are used concurrently, each of which with S colors, the direction of movement of node i at iteration k is given by

$$\theta_i(k) = 2\pi \sum_{p=1}^{P} S^{-p} c_{ip}(k) \tag{4.10}$$

resulting an an accuracy of $2\pi S^{-P}$. For any $\epsilon < 2\pi$, we have that $2\pi S^{-P} < \epsilon$ implies

$$P \geq \left\lceil -\frac{\log\left(\frac{\epsilon}{2\pi}\right)}{\log S} \right\rceil = -\left\lceil \frac{\log\left(\frac{\epsilon}{2\pi}\right)}{\log S} \right\rceil + 1. \tag{4.11}$$

4.7 Communication Architecture: Packet-Switched or Cross-Layer?

We introduce two communication architectures for implementing the proposed algorithm. In both cases, nodes access the medium at random and asynchronously; however, the signal transmission and reception models are quite different.

4.7.1 Packet-Switched Architecture

This method is based on the use of packetized data using the standard random access protocol stack which means the algorithm runs at the application layer. The proposed algorithm can be implemented easily on commercial wireless platforms like TinyOS [33] in an event-driven fashion. Only two events need to be handled at the node level- the scheduling of local transmissions and message receptions from the neighborhood. In Figure 4.6 is shown an example in pseudo-code. Let λ be the average frequency with which the node fires, and therefore, the average frequency of an update. The average time between updates is $1/\lambda$, and this parameter should be chosen so that the mechanical control can converge during this average time, but also it should be small enough to avoid the network becoming disconnected, in the case that the mechanical dynamics of the nodes include spatial motion in addition to heading alignment. The specifics of incorporating spatial control into such a scenario for purposes of enabling network connectivity is beyond the scope of this thesis and is an area of open and ongoing research. The reader is referred to [43, 45, 58] for some related work.

In this context, by using the technique based on zooming levels explained in

the previous section, the total number of bits required to form a message is then given by

$$P\lceil \log_2 S \rceil = P\left\lceil \frac{\log S}{\log 2} \right\rceil \simeq -\frac{\frac{\epsilon}{2\pi}}{\log 2} + \frac{\log S}{\log 2}.$$

We can see that the case with multiple instances of the 2-colors algorithm is optimal in terms of number of bits and time to convergence. Thus, in the case of packet-switched architectures using a codeword is preferable, given the packet overhead. This scheme is also useful in the case of *data driven* architectures, as it will be clear. Therefore, we will use this approach in the following implementations. In the most simple case we can use 2 colors, and, thus, a sequence of $P = -\left\lceil \frac{\log\left(\frac{\epsilon}{2\pi}\right)}{\log 2} \right\rceil + 1$ bits to encode the message.

The advantage of using the packet-switched architecture is that it is standard and easy to build. Unfortunately, it is not scalable to networks with a large number of nodes and dense neighborhoods because of interference. Status updates in dense neighborhoods lead to channel contention and interference which in turn reduce the speed of convergence [27]. This issue is exacerbated when the communication media is harsh, as in underwater environments.

4.7.2 Cross-Layer Architecture and Application to Underwater Environments

The coordinated multi-vehicle motion control is not limited to land or air environments. One such environment/applications is underwater robotics. For instance, Figure 4.7 shows a picture of the Fin-Actuated Autonomous Underwater Vehicle, developed by Morgansen's Research Group at the University of Washington,

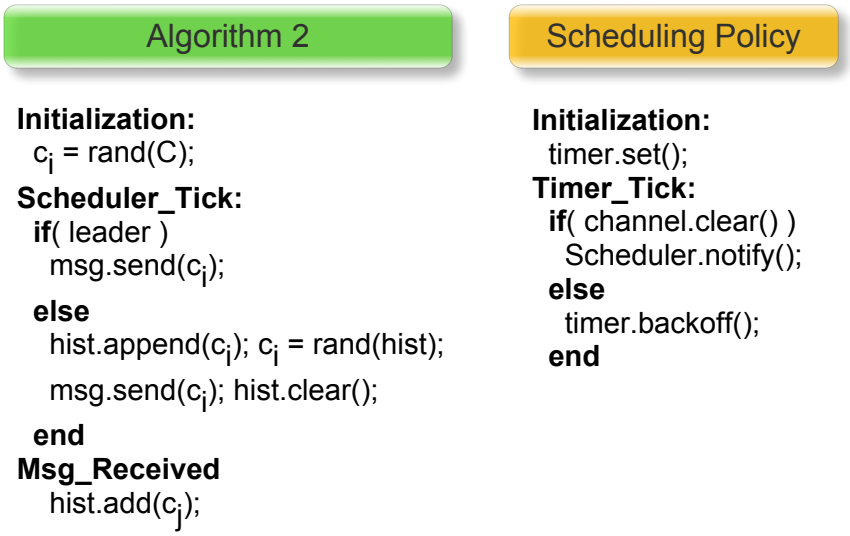

Figure 4.6: Example of implementation of Algorithm 2, suitable for event-driven wireless sensor platforms, such as TinyOS.

Seattle [44]. Unfortunately, underwater acoustic channels are a challenging media for communication [67, 79], rendering the packetized solutions inefficient and impractical.

Recall, in a packet-switched multiple access channel, communication resources are allocated on a per node basis. For purposes of illustration, suppose FDMA is used. Let B_u denote the allocated bandwidth per user. In any round, each node needs to transmit once. The use of guard intervals to mitigate Doppler effects will require an *average* of $2 \times [(190 - 50)/2 + 10] = 160Hz$ per node [79]. Assuming the network has n nodes, B_u must satisfy the condition: $n(B_u + 160) \leq B_a$, where B_a is the total available bandwidth. Since $B_u > 0$, if $B_a = 10kHz$ (this is the usable frequency band when communicating over a distance of $5km$), this condition either limits the network size to $n < 63$, or for a given n, it limits the bandwidth to $B_u \leq B_a/n - 160Hz$, and hence the rate available to each node.

Motivated by these considerations, we propose an alternative cross-layer architecture that is better equipped to handle the challenges of underwater communications and other resource constrained communication environments. One might notice, that perhaps a simple measure of *occupancy* of a certain color might work as well. Such a measure should satisfy the following properties: (i) if a color is not in use then all nodes should record zero occupancy for that color and (ii) if a color is in use and it is broadcast by a node, then some node in the network should be able to record a positive occupancy for that color. Capitalizing on these observations, we have designed a second method which is cross-layer. We argue that this approach is preferable in harsh media where the traditional protocol stack can be inefficient.

Instead of sending its color via data packets, each node emits a *whistle* at a frequency that is mapped uniquely to its color. Like before, given that there are $|\mathcal{V}|$ available frequencies, we divide this set into disjoint subsets of two channels each, so that the algorithm becomes the composition of $P = |\mathcal{V}|/2$ concurrent instances, each representing a different zooming level. Therefore, in the following we will describe the case where two channels are used by the nodes, keeping in mind that this this task is performed in parallel by using a filter-bank and FFTs. Let $g(\cdot)$ be a one-to-one mapping from each of the P independent 2-color instances of our algorithm to the set $\mathcal{V}$ of available channels. For each of those, the node chooses one of the two channels (based on Eq. (4.10)), say ν_l, and broadcasts a continuous signal $v_l(t) = \xi \cos(2\pi(f_o + \frac{B_o}{|\mathcal{V}|}\nu_l)t + \phi_l)$, where ξ is the signal amplitude, f_o is a base carrier frequency, and $\phi_l \overset{iid}{\sim} U[0, 2\pi]$ is a random phase.[8] In addition to assumptions A.1, A.2 and A.3, we require the $|\mathcal{V}|$ channels to be mutually orthogonal.

[8]For clarity, we indicate with $g(c_i)$ the particular frequency chosen by node i in a particular zooming level, out of the available $P = |\mathcal{V}|/2$.

Figure 4.7: University of Washington Fin-Actuated Autonomous Underwater Vehicle operating autonomously using data transmitted via wireless RF transceiver. On-going activities are focused on the deployment of the WHOI micro-modem for acoustic communications.

The impulse response of the underwater multipath channel between two nodes i and l is modeled by $h_{il}(t) = \sum_k h_{il}^k(t - \tau_k)$ where $h_{il}^k(t)$ is the inverse Fourier transform of the frequency response of the kth path, which is a function of attenuation and reflection coefficient [79]. Let $H_{il}(f)$ be the frequency response of the channel between nodes i and l. We assume that the channels between distinct node-pairs are independent. In the $1 - 100kHz$ communication range, the dominant source of noise is ambient surface noise which can be written as $N(f) = NL_{1K} - 17\log(\frac{f}{1000})$ using Knudsen's model, where NL_{1K} is a constant dependent on the sea condition [51]. Since the signals $v_j(t)$ transmitted in the cross-layer architecture occupy

narrowbands even after accounting for typical Doppler effects, we assume that the noise spectrum in the jth frequency bin centered at f_j is flat $N(f_j) \approx \sigma_j^2$. We assume the use of extremely simple radio devices that are non-coherent *i.e.* neither the transmitter nor the receiver have channel state information, and that the nodes are half-duplex and transmit the same average power.

Whenever $\mathcal{I}(k) = i$, node i stops transmitting on its channel and begins receiving. Let $t(u_i^{(s)})$ be the time of the sth updating event of node i. Omitting the time dependence for clarity, at frequency ν_j, node i receives the signal

$$r_i[\nu_j] = \sum_{l \in \mathcal{L}} H_{il}(\nu_j) \xi \delta[\nu_j - g(c_l)] + w_i[\nu_j], \tag{4.12}$$

where $\mathcal{L}$ denotes the set of nodes that are active during node i's reception period that lasted until $t(u_i(s))$ and $w_i[\nu_j]$ is the ambient noise at the output of the filter.

Instead of a histogram of the received colors as in the packet-switched case, node i now receives 2 analog values representing the sums of contributions from its neighbors. Let us define

$$P_i[\nu_j] \;\; = \;\; \left(|r_i[\nu_j]|^2 - T(\sigma_j^2)\right)^+, \tag{4.13}$$

where $(\cdot)^+ = \max\{0, \cdot\}$ and $T(\sigma_j^2)$ is some threshold that depends on the noise power. Node i measures the power $|r_i[\nu_j]|^2$ received on each of the two frequencies ν_1 and ν_2 by estimating the power spectrum of the received signal during the time window of reception via a standard periodogram. The new channel node i then chooses for transmission is selected according to the probability density function

$$\mathcal{P}(\nu) = p_i[\nu_1]\delta(\nu - \nu_1) + p_i[\nu_2]\delta(\nu - \nu_2), \tag{4.14}$$

where $p_i[\nu_j] = \frac{P_i[\nu_j]}{\sum_{l=1}^{2} P_i[\nu_j]}$. The updating rule induced by (4.14) is equivalent to the case of selecting a color based on neighbors' states. The only difference is that, in

this case, instead of colors, we have power levels associated to different orthogonal channels.

Remark 1 *Note that, from* (4.12)*, even if nobody is transmitting on a particular channel* ν_j*, there is a nonzero probability that node* i *receives some power due to the presence of noise whose value can exceed the threshold* $T(\sigma_j^2)$*. Suppose that the network is at consensus on channel* $\nu[l']$ *after* k *rounds. Denote this by event* $\mathcal{C}_k$*. Define the event* $\mathcal{E}_i = \{|r_i[\nu_l]|^2 \leq T(\sigma_l^2), \forall l \neq l'\}$*, i.e., the power received by node* i *on channels other than the one in which it is transmitting is below the threshold. Recalling that the nodes use two channels for each partition, and exploiting the independence of the signals received by each node, the probability that the network will stay in consensus after* n *consecutive updates is lower-bounded by*

$$\begin{aligned} P(\mathcal{C}_{k+1}|\mathcal{C}_k) &\geq P(\mathcal{E}|_{i=1}^n) \\ &= P(|r_i[\nu_l]|^2 \leq T(\sigma_l^2))^n = \left(1 - e^{-\frac{T(\sigma_l^2)}{\sigma_l^2}}\right)^n. \end{aligned} \tag{4.15}$$

Setting the right-hand side to ϕ *implies that* $T(\sigma_l^2) \geq -\sigma_l^2 \ln(1 - \phi^{\frac{1}{n}})$ *guarantees* $P(\mathcal{C}_{k+1}|\mathcal{C}_k) \geq \phi$*. We can see that, as* $T(\sigma_l^2)$ *increases,* $P(\mathcal{C}_{k+1}|\mathcal{C}_k) \to 1$*. However, if* $T(\sigma_l^2)$ *is too big, the nodes will become disconnected, so there is a trade-off between robustness and connectivity. In practice, having* $T(\sigma_l^2) = a\sigma_l^2$*, with* $a \sim 3$ *provides good results.*

Remark 2 *The advantage of the cross-layer approach is that control information for channel access and for error correction is completely eliminated, resulting in a streamlined architecture for the transmission. In fact, in this architecture there is no need of scheduling for the MAC or contention or CSMA. This is because the nodes that have the same states, proportionally increase the received power only*

over that corresponding channel, and there is no need to separate those transmissions. At the physical layer, the spectrum estimation includes the effect of fading and linear distortion as a scaling, and since phase information is irrelevant, complex training and synchronization will not need to be performed.

In the case of Algorithm 2 (ALIGNMENT WITH LEADER), the time to convergence can be further reduced by requiring the leader to use a larger transmission power. The Leader spends more energy than the others, and the choice of these alternatives depend on the specific energy budget.

4.8 Numerical Results

In this section we provide illustrative examples and numerical results regarding the performances of the algorithms we have presented. As discussed in 4.6, we consider both Algorithms 1 and 2 with 2 colors, since the nodes may run multiple instances and retrieve the direction using equation (4.10).

In Figure 4.8 is shown the direction of the nodes over time. The network is a connected random geometric graph with $n = 25$ nodes, where two nodes i and j can communicate if their distance $d(i, j)$ is less than $r = 0.12$. In this specific case, 6 instances of Algorithm 2 are running concurrently, resulting in an accuracy of 5.6 degrees. We can see that the nodes reach consensus to the direction of the leader in about 600 iterations, corresponding to 24 iterations per node, corresponding to about n updates per node, as expected.

Figures (4.9)-(4.10) show the number of iterations per node needed to converge, for both Algorithm 1 and 2, with transmission radius ($r = \sqrt{4n^{-1} \log n}$). In these

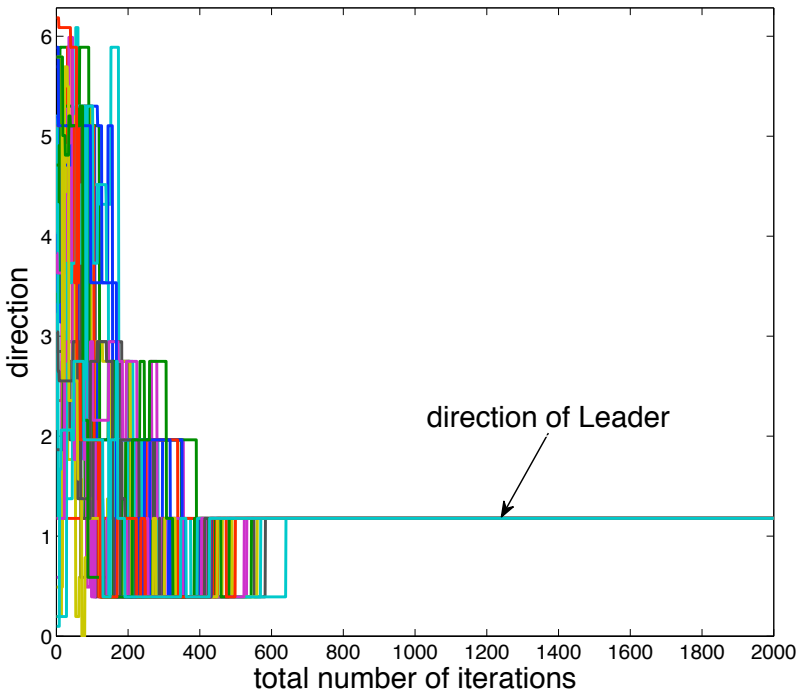

Figure 4.8: Packet switched architecture: evolution of a network composed of $n = 25$ nodes with normalized radius $r = 0.12$. Each node encodes its direction using 6 bits, therefore performing 6 instances of Algorithm 2, with 2 colors, concurrently (5.6 degrees of accuracy). The network reaches consensus to the direction of the leader in about 600 iterations, corresponding to 24 iterations per node.

two figures are reported 100 experiments for each value of n. Furthermore, the squares in both figures indicate the mean value, averaged over 10^5 experiments. These results suggest that the upper-bounds we derived above hold. In particular, we can say that a linear trend is sufficient to bound the number of iterations needed per node to converge, in the two-colors case. Since any level of desired accuracy can be achieved with a sufficient number of binary instances of our coloring algorithm, we can conclude that the algorithm we proposed converges in linear time with high probability.

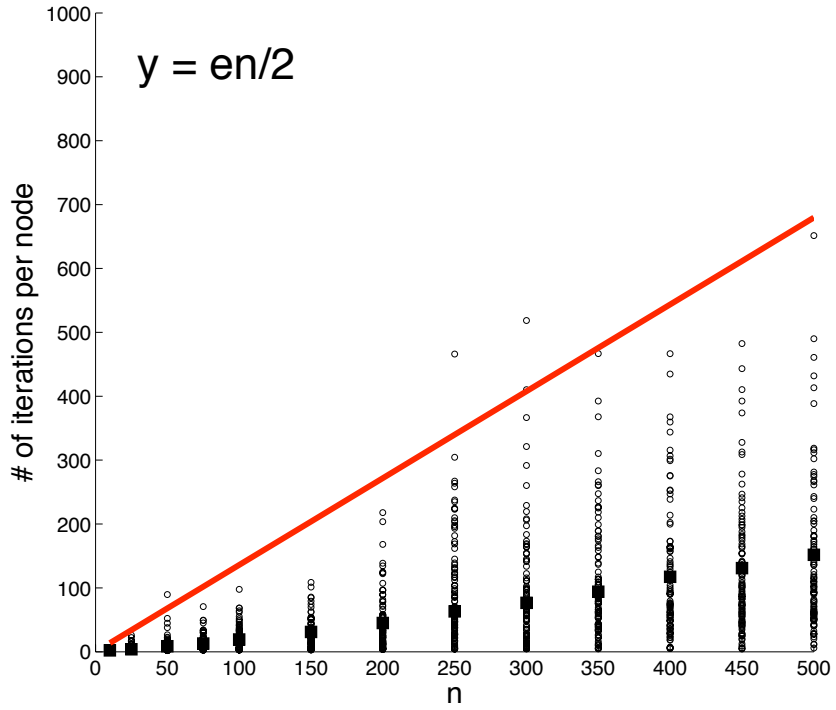

Figure 4.9: Algorithm 1: number of iterations per node with constant connectivity ($r = \sqrt{4n^{-1}\log n}$). The linear bound is indicated in red, and the squares are the averages for each value of n. [100, 99.9, 99.8, 99.7, 99.7, 99.7, 99.6, 99.6, 99.6, 99.5, 99.4, 99.4, 99.4].

In Figure 4.11 is reported the evolution of a network composed of $n = 50$ nodes within a 25-by-25 square area (in arbitrary units). The direction of the leader is approximatively 180 degrees and the threshold used by the nodes is $T(\sigma^2) = 5\sigma^2$. As expected, the nodes are driven toward the direction of the leader. We note that a few nodes go temporarily out of consensus, due to noise, as discussed in Remark 1. In the same picture is also reported, for each node, the average SNR (in dB) received by the others.

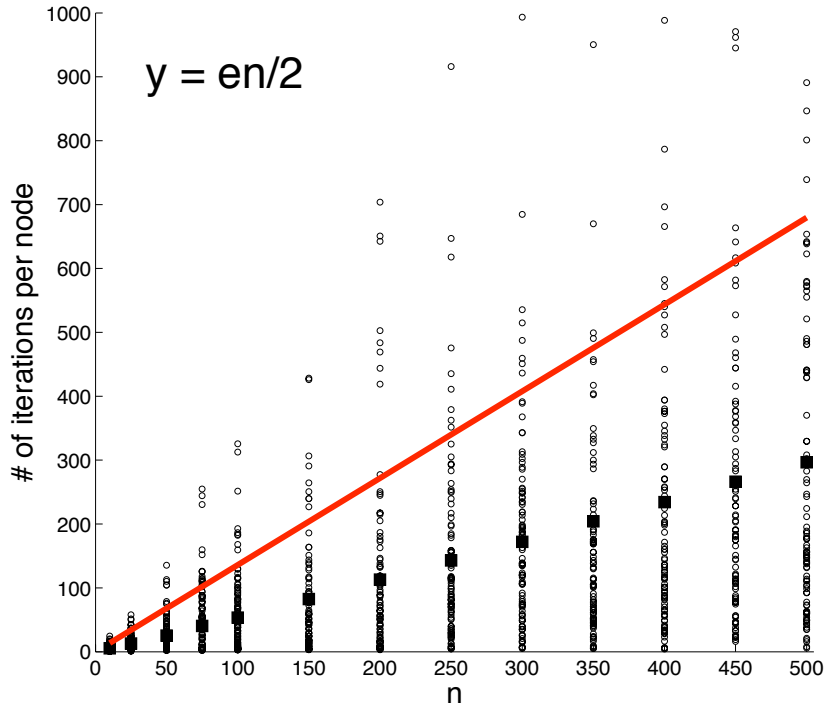

Figure 4.10: Algorithm 2: number of iterations per node with constant connectivity ($r = \sqrt{4n^{-1}\log n}$). The linear bound is indicated in red, and the squares are the averages for each value of n. [91.8, 91.5, 91.4, 90.5, 90.7, 90.6, 90.4, 90.6, 90.7, 90.4, 90.3, 90.1, 90.2].

4.9 Voting on a Graph: Disagreement

As we have seen from the specifications of the model described in Section 4.3, the nodes are always driven to consensus by means of local interactions. One might ask what happens if the updating rule is changed. After all, Eq. (4.2) simply means that the updating node copies the color of one of its neighbors, chosen at random. The vote is biased towards becoming the vote of the majority, but is not necessarily that of the majority. Hence, a straightforward modification of such dynamics is the majority rule, also known as Label Propagation Algorithm (LPA) [72]. This

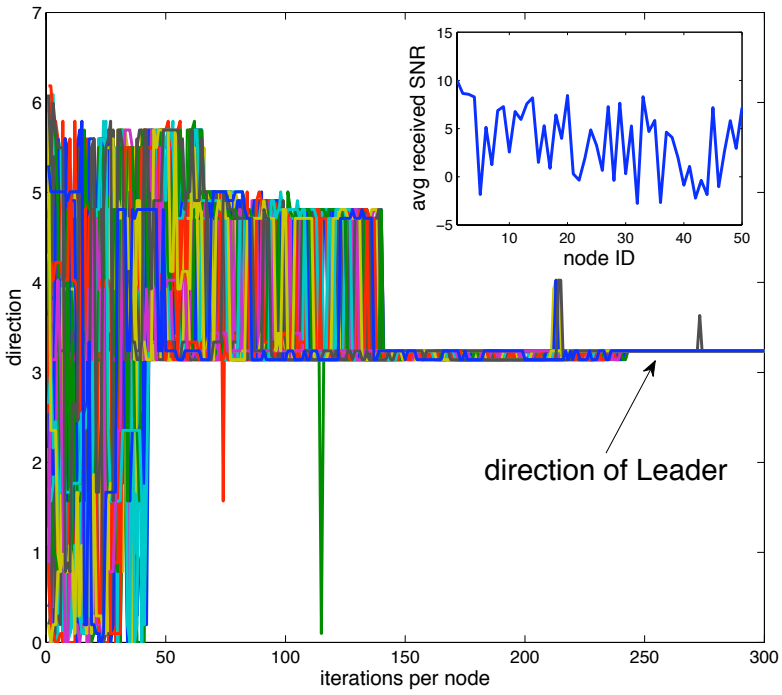

Figure 4.11: Cross-layer architecture: evolution of a network composed of $n = 50$ within a 25-by-25 square area (a.u.). Each node encodes its direction using 6 channels, therefore performing 6 instances of Algorithm 2, concurrently (5.6 degrees of accuracy). In the inset is reported the average SNR of each node received by the others.

particular scheme is somewhat similar to the one we discussed above. Each node wakes-up and collect the colors of its neighbors. Instead of picking one at random, the node chooses the color with the most number of occurrences (hence, *majority* rule). Suppose node i wakes up at iteration k. Recalling that $Q_{i,c}[k]$ is the number of neighbors of node i colored with c, the new state of node i is given by the following updating rule

$$c_i[k+1] = \arg\max_c Q_{i,c}[k]. \tag{4.16}$$

If two or more colors satisfy Eq. (4.16), node i chooses one of such colors at random. We can see that, as opposed to Eq. (4.2), the updating rule is deterministic,

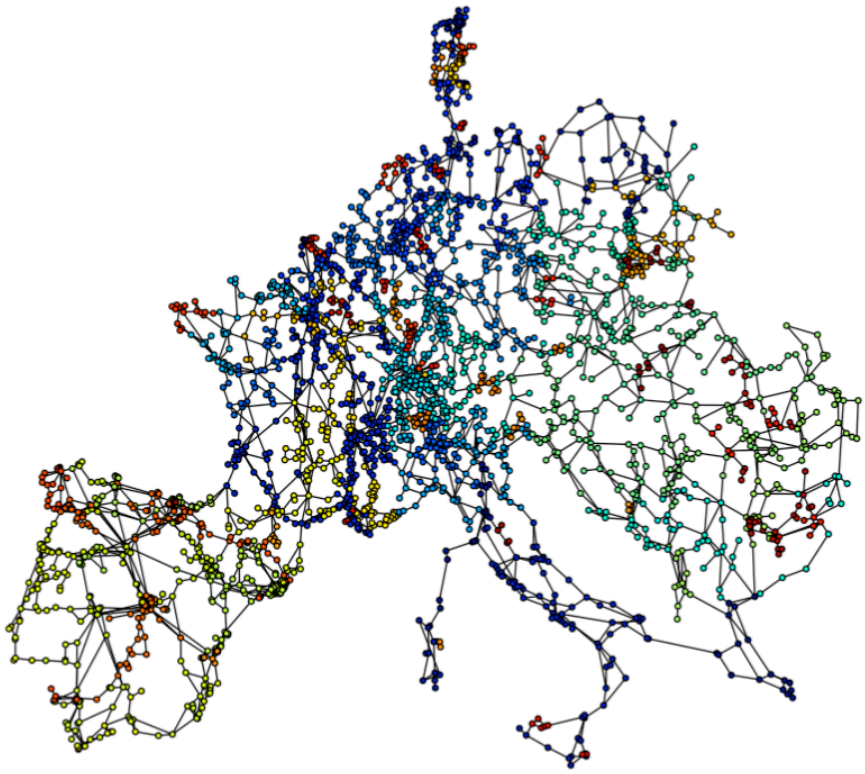

Figure 4.12: Label Propagation Algorithm (LPA) applied to the European Power Grid.

since the node chooses the opinion adopted by the majority of its neighbors. The LPA algorithm does not lead the nodes to consensus, nor (in general) guarantees convergence. However, an interesting pattern emerges as the node keep updating their colors using Eq. (4.16). The number of colors reduces, and nodes within a community end up sharing the same color. By community in a graph, we mean a subgraph of G such that the number of edges within the subgraph is much bigger than the number of edges connecting the subgraph to the rest of the network. Alternatively, one may define the concept of community in a graph as a region where the number of links is much bigger with respect to a reference graph (the so called *null model*), corresponding to what we would expect by adding edges between nodes at random [60].

In Figure 4.12 we show the result of the LPA algorithm applied to a large graph, the European Power Grid. We can see that "dense" regions of nodes are identified by one color. Although there are not many theoretical results about the properties of LPA, the algorithm seems to be working quite well with many graphs of interest. Potential applications of this scheme include self-organization for mobile robots and power control in wireless networks.

The LPA algorithm has been improved quite significantly since its discovery. In [48] an modified version of the LPA for finding communities of similar size has been proposed. In [26] the Authors designed an LPA-based algorithm to detect the extent to which a node belongs to one of the surrounding communities. An interesting feature offered by that alternative is this weighted and bipartite networks are handled as well. In [81] it has been shown that the LPA algorithm is equivalent to finding the local minima of a simple Potts model.

The Label Propagation Algorithm has also potential applications in the engineering panorama. This scheme can, in fact, be used for formation control problems. In this context, one of the main problems is keeping the network "well" connected over the time; this means that the formation of communities is not desirable. If the nodes perform the LPA scheme, some of them will know that they lie over the borders between different communities, allowing them to inform the others to get closer. The same idea can be employed to improve the connectivity of low-density networks.

CHAPTER 5

CONCLUSIONS

In this thesis we studied two main classes of biologically inspired algorithms, namely, the Pulse-Coupled Oscillator and Nonlinear Voting models. These protocols, driven by the beauty of simplicity of biological systems, showcase a better scalability and more degree of adaptability if compared to traditional protocols for diffusion of information.

In Chapter 2 we derived novel theoretical results on the convergence of the PCO protocol and showed through experiments its efficacy. Our results indicate that the PCO algorithm outperforms other competing distributed schemes, such as one of its previous implementation and the RBS, and, therefore, represents a valid alternative in the context of synchronization protocols for sensor networks

In Chapter 3 we extended the classic PCO model, originally proposed by Peskin, to the case with inhibitory coupling. In this case, rather than synchronizing, the nodes participating in the algorithm converge to a state where their clocks separate in time of a constant quantity, a condition which is called desynchronization. However, this scheme is not directly implementable in wireless devices, since it is not robust to non-idealities in the communication channel. Hence, we proposed novel schemes to overcome this difficulty, showing that, by restricting the coupling to a subset of nodes, it is possible to achieve distributed time-sharing in a more robust way, finally discussing its implementation at the physical layer.

In Chapter 4 we introduced a novel coloring algorithm for consensus in decentralized systems. This protocol can be used as a primitive for reaching consensus in a distributed fashion, with applications to formation control of mobile robots. The

proposed algorithm offers advantages with respect to others: it does not require synchronization, nor complete knowledge of the network topology at the local level. Moreover, mobility and message drops are naturally handled by the probabilistic nature of the protocol.

The main results presented in this thesis are available in the journal publications reported below.

- **R. Pagliari**, S. Kirti, K. A. Morgansen, T. Javidi, and A. Scaglione, "A Simple and Scalable Algorithm for Alignment in Broadcast Networks", (to appear) IEEE Journal on Selected Area in Communications, Special Issue on Simple Wireless Sensor Networking Solutions, 2010.
- **R. Pagliari**, Y.-W. P. Hong and A. Scaglione, "Bio-Inspired Algorithms for Decentralized Round-Robin and Proportional Fair Scheduling", IEEE Journal on Selected Area in Communications, Special Issue on Bio-Inspired Networking, vol. 28, no. 4, 2010.
- **R. Pagliari** and A. Scaglione, "Scalable Network Synchronization with Pulse-Coupled Oscillators", (to appear) IEEE Trans. on Mobile Computing, 2010.

APPENDIX A

APPENDIX OF CHAPTER 2

A.1 Proof of Theorem 1

We first notice that all players having the same color is the only absorbing state. The state of the system, which is the vector of the players' colors, is a random walk induced by the wake up times of the players. Assume that a set of players $\mathcal{I}$ wakes up at time t_k. The transition probability of a player $j \notin \mathcal{I}(t_k)$ is, precisely, equal to $p_{j,\mathcal{I}(t_k)}(t_k)$. Since $p_{j,\mathcal{I}(t_k)}(t_k) > 0,\ \forall i, j, k$ there always exists a positive probability that group of nodes gets bigger from iteration to iteration. Since this is true for any color appearing at time $t = 0$, the result of the theorem follows.

A.2 Proof of Theorem 2

We consider a group of nodes firing at time t_k, corresponding to the k^{th} firing event. We denote this particular group with the set $\mathcal{S}$. Some of the nodes in the set $\mathcal{N} \setminus \mathcal{S}(t_k)$ detect the message, some others do not. We want to show that there exists a nonzero probability that, the next time this group of nodes broadcasts, its cardinality is bigger. We recall that the number of different color in the network can increase: in fact, some nodes may detect the message sent by the nodes in $\mathcal{S}(t_k)$ and update their state without being absorbed, meaning that they pull their state closer to them, but not exactly equal to it. Assume that the next broadcasting time of the nodes in the set $\mathcal{S}$ occurs at time t_{k+m} for some $m > 0$. At time t_{k+m} the number of nodes in $\mathcal{S}$ may increase because of the following possible reasons. 1) Some nodes get absorbed at iteration k, based on the coin toss experiment,

and none of the nodes in $\mathcal{S}(t_k^+)$ decodes the messages sent by the others between iterations $k+1$ and $k+m-1$, and this may occur since $0 < p_{j,\mathcal{I}(t)}(t) < 1,\ \forall j, \mathcal{I}, t$. 2) No new nodes are absorbed at iteration k but there exists $m' > m$ such that new nodes are absorbed in $\mathcal{S}(t_{m'})$ (these nodes keep moving toward the color of the ones in $\mathcal{S}$ eventually being absorbed). This is possible when, for example, a some nodes make a bigger number of shifts towards the color of $\mathcal{S}$ rather than any other color in the network. 3) Some nodes in $\mathcal{S}(t_k^+)$ go out of synch between iterations k and $k+m$, but a bigger number of nodes not in $\mathcal{S}(t_k^+)$ gets the same color of the nodes in $\mathcal{S}(t_k^+)$. This is possible by construction: in fact we could impose that some non firing nodes at iteration $k+m'$ (with $m' < m$) get a color which is equal to the color of the nodes in $\mathcal{S}(t_k^+)$ and go back in time until iteration $k = 0$ to find the corresponding initial condition. 4) A combination of the cases above. Therefore, given a set of nodes $\mathcal{S}(t_k)$ with the same color (synchronized nodes) at time t_k, there always exists a nonzero probability and $k' > k$ such that $\mathcal{S}(t_{k'})$ is bigger. This probability is, in general, nontrivial to compute and can be quite small, especially if the number of nodes in the network is large. However, this quantity is not zero. Therefore, the condition of agreement, which is the absorbing state, is always reachable, which is a sufficient condition to guarantee the convergence of this algorithm.

APPENDIX B

APPENDIX OF CHAPTER 3

The strategy we use to prove convergence results can be briefly described as follows. Instead of considering the convergence of each phase difference separately, we look specifically at the convergence of the entire system state $\mathbf{\Delta}[k]$. By defining the fixed point of the system state as

$$\mathbf{\Delta}^* = \lim_{k\to\infty} \mathbf{\Delta}[k] = [\Delta_n^*, \ldots, \Delta_1^*]^T, \tag{B.1}$$

our goal is to show that, $\forall \epsilon > 0$, $\exists k^*$ such that

$$||\mathbf{\Delta}[k] - \mathbf{\Delta}^*||_1 < \epsilon \tag{B.2}$$

for all $k > k^*$, where $\| \cdot \|_1$ is the 1-norm. By considering the 1-norm, Eq. (B.2) can be viewed as a bound for the sum of distances with respect to the fixed point, which also implies ϵ-convergence of $\Delta_1[k]$. In particular, if $\mathbf{\Delta}^* = c_{\text{strict}}\mathbf{1}_n$ (all elements are equal) we have strict desynchronization, in which case $c_{\text{strict}} = \frac{1}{n}$, since $||\mathbf{\Delta}^*||_1 = 1$. If $\mathbf{\Delta}^* = \mathbf{v}$ with $\frac{\mathbf{v}}{||\mathbf{v}||_1} \neq \mathbf{1}_n$, instead, we have weak desynchronization with $c_{\text{weak}} = \Delta_1^*$. Equation (B.2) will be used in the following to prove convergence and rate of convergence.

B.1 Proof of Theorem 3

In this case, the firing of a node causes all the others to perform the update. We shown next that the evolution of the state vector, from round to round, can be described by an affine transformation. Consider for example, the case with three nodes, and assume that the initial offsets, at round 0, are $\phi_3(0) > \phi_2(0) > \phi_1(0)$.

Note that, $\Phi_3(t_3) = 1$, $\Delta_3(t_3) = 1 - \Phi_2(t_3)$ while $\Delta_2(t_3) = \Phi_2(t_3) - \Phi_1(t_3)$ and $\Delta_1(t_3) = \Phi_1(t_3) - \Phi_3(t_3) \bmod 1 \ = \Phi_1(t_3)$. Considering equation (3.8), we obtain that $\Delta_3(t_3^+) = 1 - (1-\alpha)\Phi_2(t_3) = 1 - (1-\alpha)(1 - \Delta_3(t_3))$ and $\Delta_2(t_3^+) = (1 - \alpha)\Phi_2(t_3) - (1-\alpha)\Phi_1(t_3)$. Hence, when node 3 fires at time t_3, the state of the system becomes linear:

$$\begin{aligned}
\Delta_3(t_3^+) &= (1-\alpha)\Delta_3(t_3) + \alpha \\
\Delta_2(t_3^+) &= (1-\alpha)\Delta_2(t_3) \\
\Delta_1(t_3^+) &= (1-\alpha)\Delta_1(t_3)
\end{aligned}$$

At time $t_2 > t_3$ node 2 fires, and, thus, the intervals between the nodes change as $\Delta_3(t_2^+) = (1-\alpha)\Delta_3(t_2)$, $\Delta_2(t_2^+) = (1-\alpha)\Delta_2(t_2) + \alpha$ and $\Delta_1(t_2^+) = (1-\alpha)\Delta_1(t_2)$. At time $t_1 > t_2$ node 1 fires, and the state of the system becomes

$$\begin{aligned}
\Delta_3(t_1^+) &= (1-\alpha)\Delta_3(t_1) = (1-\alpha)^3\Delta_3(t_3) + (1-\alpha)^2\alpha \\
\Delta_2(t_1^+) &= (1-\alpha)\Delta_2(t_1) = (1-\alpha)^3\Delta_2(t_3) + (1-\alpha)\alpha \\
\Delta_1(t_1^+) &= (1-\alpha)\Delta_1(t_1) + \alpha = (1-\alpha)^3\Delta_1(t_3) + \alpha.
\end{aligned}$$

The evolution of the system from round R to round $R+1$ is then $\mathbf{\Delta}[R+1] = \mathbf{M}_3\mathbf{\Delta}[R] + \mathbf{v}_3$, where $\mathbf{M}_3 = (1-\alpha)^3\mathbf{I}$ and $\mathbf{v}_3 = [(1-\alpha)^2\alpha, (1-\alpha)\alpha, \alpha]^T$. More generally, if we have n nodes the system evolves as $\mathbf{\Delta}[R+1] = \mathbf{M}\mathbf{\Delta}[R] + \mathbf{v}$ where $\mathbf{M} = (1-\alpha)^n\mathbf{I}$ and $\mathbf{v} = \alpha[(1-\alpha)^{n-1}, (1-\alpha)^{n-2}, \ldots, (1-\alpha), 1]^T$. Therefore, at round $R+1$

$$\begin{aligned}
\mathbf{\Delta}[R+1] &= \mathbf{M}^R\mathbf{\Delta}[0] + \sum_{i=0}^{R-1}\mathbf{M}^i\mathbf{v} \\
&= \mathbf{M}^R\mathbf{\Delta}[0] + (\mathbf{I}-\mathbf{M})^{-1}\left(\mathbf{I}+\mathbf{M}^R\right)\mathbf{v}. \qquad \text{(B.3)}
\end{aligned}$$

$\mathbf{M}$ is a contraction, and, thus, from Eq. (B.3) we have $\mathbf{\Delta}^* = \lim_{R\to\infty} \mathbf{\Delta}[R] = (\mathbf{I} - \mathbf{M})^{-1}\,\mathbf{v}$. By expanding (B.3) we obtain

$$\begin{aligned} ||\mathbf{\Delta}[R] - \mathbf{\Delta}^*|| &= ||\mathbf{M}^R\mathbf{\Delta}[0] + (1-(1-\alpha)^n)^{-1}\mathbf{M}^R\mathbf{v}|| \\ &= (1-\alpha)^{nR}\,||\mathbf{\Delta}[0] + (1-(1-\alpha)^n)^{-1}\mathbf{v}|| \\ &\leq 2(1-\alpha)^{nR} < \epsilon. \end{aligned} \tag{B.4}$$

Condition (B.4) implies $R^* = -\frac{1}{n}\frac{\log\left(\frac{2}{\epsilon}\right)}{\log(1-\alpha)}$; whenever this quantity becomes less than 1 (number of firings less than n), we set $R^* = 1$ and the result of the theorem follows. The spacing between two consecutive firings is given by $\Delta_1^* T_f = ((\mathbf{I} - \mathbf{M})^{-1}\mathbf{v})_1 T_f = \frac{\alpha}{1-(1-\alpha)^n} T_f$.

The spacing between two consecutive rings converges to a constant with value $\Delta_1 T_f = \frac{\alpha}{1-(1-\alpha)^n} T_f$. This shows that, although the phase difference between any two nodes do not converge to a fixed value throughout the process, this scheme is sufficient to achieve round-robin scheduling where each node gets equal share of the channel. Interestingly, as $\alpha \to 0$, the spacing between two rings is converges to $1/n$, which is the value achieved in the case of strict desynchronization. However, as $\alpha \to 0$ the time to convergence increases as $1/\alpha$.

B.2 Proof of Theorem 4

Without loss of generality, we assume that $\Delta_i(0) > 0, \forall i = 1, 2, \ldots, n$.[1] We first show that if two consecutive nodes are separated by a gap strictly bigger than zero, then, those nodes can never collapse into the same point. If this statement does not hold, then it implies that $\lim_{t\to\infty} \Delta_j(t) = 0$ for some j. This implies

[1] In fact, if the nodes choose their initial phases at random and independently, the probability that any two nodes pick exactly the same value is zero.

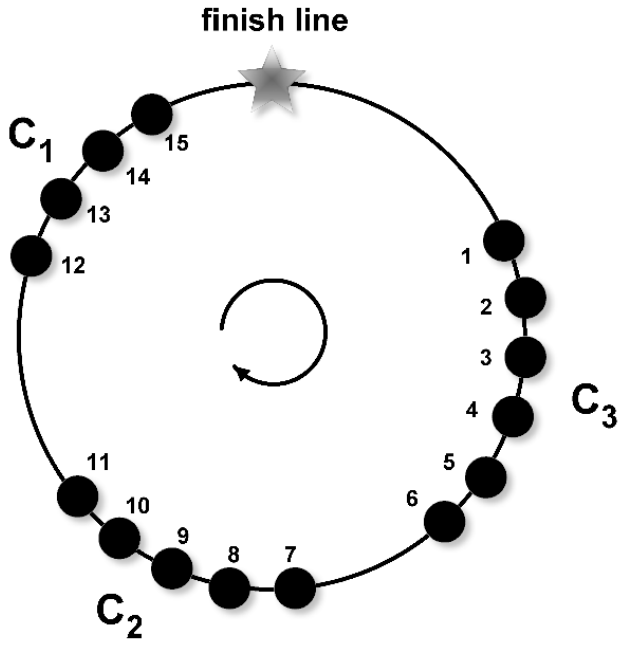

Figure B.1: Example of chains of nodes at time $t > 0$ in a network composed of $n = 15$ nodes. C_1, C_2 and C_3 are the three chains of the system at time t.

that $\lim_{t\to\infty} \Phi_{j-1}(t) = \Phi_j(t)$. But from (3.9), when $\Phi_{j-1}(t) \to \Phi_j(t)$ the firing of node j causes node $j-1$ to update as $\Phi_{j-1}(t^+) \to 1 - \frac{\alpha}{n} \neq 0$. Therefore, $\Delta_j(t_j^+) = \alpha/n \neq 0$, leading to a contradiction.

Prior to proceeding, we define a *chain* of nodes at time t as a set of consecutive nodes whose inter-distances are less or equal to $1/n$. An example is shown in Figure B.1 where, at time t, three chains are present in the system. We indicate with $\mathcal{C}(t)$ the set of chains in the system at time t, and its cardinality (the number of chains) with $|\mathcal{C}|$. In Figure B.1, for instance, $\mathcal{C}(t) = \{C_1, C_2, C_3\}$ and $|\mathcal{C}(t)| = 3$. Moreover, we call *size* of chain C_l the distance between the two extreme nodes in the chain. In Figure B.1, for example, the size of chain C_1 is equal to $\Delta_{13}(t) + \Delta_{14}(t) + \Delta_{15}(t) =$

$\Phi_{15}(t) - \Phi_{12}(t)$. Notice that, by definition, these chains are separated by gaps bigger than $1/n$.

Lemma 5 *The cardinality of the set $\mathcal{C}(t)$ is strictly positive and non increasing.*

Proof The first result is trivial, since there always exists a pair of adjacent nodes whose distance is less than, or equal to $1/n$ and, thus, $|\mathcal{C}(t)| \geq 1$. Now, suppose that $|\mathcal{C}(t)|$ increases. This implies that a node (or a group of nodes) leaves its chain and form a new one, due to the firing of another node in the same chain. But a node never pushes apart another one more than $1/n$, leading to a contradiction. Q.E.D..

On the other hand, the cardinality of $\mathcal{C}(t)$ can decrease, and this happens if the phase-jump made by a node is such that its distance to the next one becomes less than $1/n$. Since the cardinality of $\mathcal{C}(t)$ is non increasing, and at least equal to one, the sequence of integer variables $\{|\mathcal{C}[k]|\}$ has a minimum, and this minimum is reached in finite time. Suppose k^* is the first iteration at which the minimum is reached. Then, $\forall k > k^*$ the set of chains (separated by gaps bigger than $1/n$) does not change. Because of (3.9) the size of each chain C_l increases monotonically to $\frac{C_l - 1}{n}$, while each gap decreases to $1/n$, and the result of Theorem 4 follows.

B.3 Convergence Rate of Strict Desynchronization

Recall that $n_0 = n$. We consider the special case where the firing of a node only affect its neighbor (i.e., the node firing immediately after it) and where initially $\Delta_i[0] < 1/n, \forall i = 2, \ldots, n$ and $\Delta_1[0] > 1/n$. In this case, $\Delta_1[k]$ will decrease monotonically towards $1/n$ in each round, as described in Appendix B.2, and, therefore,

$\sum_{i=2}^{n} \Delta_i[k]$ will increase towards $1 - 1/n$. Since it must hold that $\sum_{i=1}^{n} \Delta_i[k] = 1$ (which implies that $\Delta_1[k] = 1 - \sum_{i=2}^{n} \Delta_i[k]$), the system state of round k is uniquely specified by the $n-1$ dimensional vector $\hat{\mathbf{\Delta}}[k] = (\Delta_n[k], \Delta_{n-1}[k], \ldots, \Delta_2[k])^T$. The distance to the fixed point at round k is then given by

$$\begin{aligned}
\left|\left|\mathbf{\Delta}[k] - \frac{1}{n}\mathbf{1}_n\right|\right|_1 &= \left|\left|\hat{\mathbf{\Delta}}[k] - \frac{1}{n}\mathbf{1}_{n-1}\right|\right|_1 + \left(\Delta_1[k] - \frac{1}{n}\right) \\
&= \left|\left|\hat{\mathbf{\Delta}}[k] - \frac{1}{n}\mathbf{1}_{n-1}\right|\right|_1 + \left(1 - \sum_{i=2}^{n} \Delta_i[k] - \frac{1}{n}\right) \\
&= 2\left|\left|\hat{\mathbf{\Delta}}[k] - \frac{1}{n}\mathbf{1}_{n-1}\right|\right|_1 \qquad \text{(B.5)}
\end{aligned}$$

where $\mathbf{1}_{n-1}$ is the $n-1$ dimensional vector of ones. The convergence time is the minimum number of rounds required to satisfy the condition $2\left|\left|\hat{\mathbf{\Delta}} - \frac{1}{n}\mathbf{1}_{n-1}\right|\right| < \epsilon$.

To study the convergence of $\hat{\mathbf{\Delta}}[k]$, we apply the methodology used in Appendix B.1 to find the recursive update of the phase difference vector, which we can show is also affine. Let us consider the updates in the k-th round and recall that the firing of each round starts with node n. When node n fires, the phase of node $n-1$ is updated as (3.10) and, thus, the variables Δ_n and Δ_{n-1} become

$$\Delta_n(t_n^+) = (1-\alpha)\Delta_n(t_n) + \frac{\alpha}{n} \qquad \text{(B.6)}$$

$$\begin{aligned}
\Delta_{n-1}(t_n^+) &= \Delta_n(t_n) + \Delta_{n-1}(t_n) - \Delta_n(t_n^+) \\
&= \alpha\Delta_n(t_n) + \Delta_{n-1}(t_n) - \frac{\alpha}{n}, \qquad \text{(B.7)}
\end{aligned}$$

where t_i is the firing time of node i in the k-th round. Then, as node $n-1$ fires, only node $n-2$ will update its phase and, thus, the variables $\Delta_{n-1}(t_{n-1}^+)$ and $\Delta_{n-2}(t_{n-1}^+)$ can be computed as

$$\begin{aligned}
\Delta_{n-1}(t_{n-1}^+) &= (1-\alpha)\alpha\Delta_n(t_n) + (1-\alpha)\Delta_{n-1}(t_n) + \frac{\alpha^2}{n} \\
\Delta_{n-2}(t_{n-1}^+) &= \Delta_{n-2}(t_n) + \alpha\Delta_{n-1}(t_n) + \alpha^2\Delta_n(t_n) - \frac{\alpha}{n} - \frac{\alpha^2}{n}
\end{aligned}$$

Next, the firing of node $n-2$ will then modify $\Delta_{n-2}(t_{n-1}^+)$ and $\Delta_{n-3}(t_{n-1}^+) = \Delta_{n-3}(t_n)$ and so on and so forth, until node 1, whose firing will have no effect on the others, since $\Delta_1 > 1/n$. By some algebraic manipulation, the state of the network at round k can be expressed as

$$\begin{aligned}\hat{\mathbf{\Delta}}[k] &= \mathbf{M}^k\hat{\mathbf{\Delta}}[0] + \sum_{i=0}^{k-1}\mathbf{M}^i\mathbf{v} \\ &= \mathbf{M}^k\hat{\mathbf{\Delta}}[0] + (\mathbf{I}-\mathbf{M})^{-1}(\mathbf{I}+\mathbf{M}^k)\mathbf{v} \end{aligned} \tag{B.8}$$

where

$$\{\mathbf{M}\}_{ij} = \begin{cases} (1-\alpha)\alpha^{i-j}, & \text{for } i \geq j \\ 0, & \text{otherwise,} \end{cases}$$

and $\mathbf{v} = \frac{1}{n}(\alpha, \alpha^2, \ldots, \alpha^{n-1})^T$. By induction, we can show that

$$\{(\mathbf{I}-\mathbf{M})^{-1}\}_{ij} = \begin{cases} \frac{1}{\alpha}, & \text{for } i = j \\ \frac{1-\alpha}{\alpha}, & \text{for } i > j \\ 0, & \text{otherwise,} \end{cases}$$

and, thus, $(\mathbf{I}-\mathbf{M})^{-1}\mathbf{v} = \frac{1}{n}\mathbf{1}_{n-1}$. Therefore, by (B.5), the distance to the fixed point at round k can be bounded by

$$\begin{aligned}&\left|\left|\mathbf{\Delta}[k] - \frac{1}{n}\mathbf{1}_n\right|\right|_1 \\ &= 2\left|\left|\mathbf{M}^k\hat{\mathbf{\Delta}}[0] + (\mathbf{I}-\mathbf{M})^{-1}(\mathbf{I}+\mathbf{M}^k)\mathbf{v} - \frac{1}{n}\mathbf{1}_{n-1}\right|\right|_1 \\ &= 2\left|\left|\mathbf{M}^k\hat{\mathbf{\Delta}}[0] + (\mathbf{I}-\mathbf{M})^{-1}\mathbf{M}^k\mathbf{v}\right|\right|_1 \\ &= 2\left|\left|\mathbf{M}^k\hat{\mathbf{\Delta}}[0]\right|\right|_1 + 2\left|\left|(\mathbf{I}-\mathbf{M})^{-1}\mathbf{M}^k\mathbf{v}\right|\right|_1, \end{aligned} \tag{B.9}$$

where we used the fact that the norm of a product is less than the product of the norms. The terms in (B.9) can be evaluated as follows.

First of all, by the definition of $\mathbf{M}$ and $\mathbf{v}$, we have that

$$\mathbf{M}\mathbf{v} = \frac{(1-\alpha)}{n}(\alpha, 2\alpha^2, \ldots, (n-1)\alpha^{n-1})^T \prec (1-\alpha)n\mathbf{v}, \tag{B.10}$$

where $\prec$ is the element-wise inequality. By applying the bound k times, we have $\mathbf{M}^k\mathbf{v} \prec (1-\alpha)^k n^k \mathbf{v}$. Hence,

$$\begin{aligned}(\mathbf{I}-\mathbf{M})^{-1}\mathbf{M}^k\mathbf{v} &\prec (\mathbf{I}-\mathbf{M})^{-1}(1-\alpha)^k n^k \mathbf{v} \\ &= (1-\alpha)^k n^k \frac{1}{n}\mathbf{1} = (1-\alpha)^k n^{k-1}\mathbf{1}\end{aligned}$$

and the second term in (B.9) becomes

$$\left|\left|(\mathbf{I}-\mathbf{M})^{-1}\mathbf{M}^k\mathbf{v}\right|\right|_1 < (1-\alpha)^k n^k. \tag{B.11}$$

Now, consider the product $\mathbf{M}\hat{\mathbf{\Delta}}[0]$. Since $\mathbf{M}$ can be viewed as a convolution matrix, the i^{th} element of the vector $\mathbf{M}\hat{\mathbf{\Delta}}[0]$ is given by

$$\begin{aligned}\{\mathbf{M}\hat{\mathbf{\Delta}}[0]\}_i &= (1-\alpha)\sum_{j=1}^{i}\alpha^{i-j}\hat{\Delta}_j[0] < \frac{1-\alpha}{n}\alpha^i\sum_{j=1}^{i}\alpha^{-j} \\ &= \frac{1-\alpha}{n}\alpha^i\frac{\alpha^{-(i+1)}-\alpha^{-1}}{\alpha^{-1}-1} < \frac{1}{n}\alpha^i\alpha^{-n}\end{aligned} \tag{B.12}$$

where the first inequality follows from the assumption that $\hat{\Delta}_j[0] < 1/n,\ \forall j$. Therefore, $\mathbf{M}\hat{\mathbf{\Delta}}[0] \prec \alpha^{-n}\mathbf{v}$ and

$$\mathbf{M}^k\hat{\mathbf{\Delta}}[0] = \mathbf{M}^{k-1}\mathbf{M}\hat{\mathbf{\Delta}}[0] \prec (1-\alpha)^{k-1}n^{k-1}\alpha^{-n}\mathbf{v}.$$

Therefore, we have

$$\left|\left|\mathbf{M}^k\hat{\mathbf{\Delta}}[0]\right|\right|_1 < (1-\alpha)^{k-1}n^{k-1}\alpha^{-n}. \tag{B.13}$$

Hence, by (B.9), it follows that

$$\begin{aligned}\left|\left|\mathbf{\Delta}[k]-\frac{1}{n}\mathbf{1}_n\right|\right|_1 &< 2(1-\alpha)^k n^{k-1}\alpha^{-n} + (1-\alpha)^k n^k \\ &< 2[(1-\alpha)n]^k\left(1+\frac{\alpha^{-n}}{1-\alpha}\right).\end{aligned}$$

Finally, by setting $2[(1-\alpha)n]^k\left(1+\frac{\alpha^{-n}}{1-\alpha}\right) < \epsilon$, we can see that, to guarantee that the distance to the fixed point is less than ϵ, the number of rounds k must be at

least equal to R^*, where

$$R^* = \left\lceil \frac{\log \frac{\epsilon}{2\left(1+\frac{\alpha-n}{1-\alpha}\right)}}{\log(1-\alpha)+\log(n)} \right\rceil = O\left(\frac{n}{\log n}\right), \tag{B.14}$$

given that $1-\alpha > 1/n$.

B.4 Proof of Theorem 5

By using variables Γ_i and Θ_i, the state of the network characterized by the $2n$-dimensional vector $\mathbf{\Delta}(t) = (\Theta_1(t), \Gamma_1(t), \ldots, \Theta_n(t), \Gamma_n(t))^{\mathrm{T}}$. From (3.13), node $i-1$'s firing causes node i to update its two clocks which, in turn, modify variables Θ_i, Δ_i and Θ_{i+1} as follows:[2]

$$\begin{aligned}\Theta_i' = \Psi_i' &= \alpha\frac{\delta}{K_i+2\delta}(\Theta_i+\Delta_i+\Theta_{i+1}) + (1-\alpha)\Theta_i \\ &= \left[1-\alpha\frac{K_i+\delta}{K_i+2\delta}\right]\Theta_i + \frac{\alpha\delta}{K_i+2\delta}\Delta_i + \frac{\alpha\delta}{K_i+2\delta}\Theta_{i+1}\end{aligned} \tag{B.15}$$

$$\begin{aligned}\Delta_i' &= \Phi_i' - \Psi_i' \\ &= \frac{\alpha K_i}{K_i+2\delta}\Theta_i + \left[1-\alpha\frac{2\delta}{K_i+2\delta}\right]\Delta_i + \frac{\alpha K_i}{K_i+2\delta}\Theta_{i+1}\end{aligned} \tag{B.16}$$

$$\begin{aligned}\Theta_{i+1}' &= \Psi_{i+1} - \Phi_i' = (\Theta_i+\Delta_i+\Theta_{i+1}) - (\Theta_i'+\Delta_i') \\ &= \frac{\alpha\delta}{K_i+2\delta}\Theta_i + \frac{\alpha\delta}{K_i+2\delta}\Delta_i + \left[1-\alpha\frac{K_i+\delta}{K_i+2\delta}\right]\Theta_{i+1}\end{aligned} \tag{B.17}$$

while all the other variables remain the same. As done in the previous sections, we consider the state at discrete times and study its evolution. From one firing to another, the new state is obtained simply by multiplying the previous one by a matrix $\mathbf{M}_i$, equal to the identity matrix except for a 3-by-3 (positive) block centered

[2]For clarity, we omit the time index.

at position $2i-1$, induced by the coefficients in (B.15), (B.16) and (B.17). The evolution of the system can then be described as:

$$\begin{aligned} \boldsymbol{\Delta}^* &= \cdots \mathbf{M}_{n-1}\mathbf{M}_n\mathbf{M}_1\mathbf{M}_2 \ldots \mathbf{M}_{n-1}\mathbf{M}_n\boldsymbol{\Delta}[0] \\ &= \lim_{R\to\infty} \mathbf{M}^R\boldsymbol{\Delta}[0] = \mathbf{M}_\infty\boldsymbol{\Delta}[0] \end{aligned} \tag{B.18}$$

where $\mathbf{M} = \prod_{i=1}^{n} \mathbf{M}_i$ and $\mathbf{M}_\infty = \lim_{k\to\infty} \mathbf{M}^k$. Each $\mathbf{M}_i$ is a full-rank matrix and, thus, $\mathbf{M}$ is full-rank as well. Moreover, it is not hard to see that $\mathbf{M}_i$ is left-stochastic for all i, which implies that $\mathbf{M}$ is left-stochastic. Since $\mathbf{M} = \prod_{i=1}^{n} \mathbf{M}_i$ is a primitive matrix[3], we can apply the Perron-Frobenius theorem and conclude that there is a positive eigenvalue $\lambda_{\max}$ strictly greater, in magnitude, than all the other eigenvalues. Since matrix $\mathbf{M}$ is left-stochastic all eigenvalue lie within the unit disc and, thus, the fixed point is unique, if there is one. Such vector must validate (B.15), (B.16) and (B.17) for all i. It can be verified that the vector

$$\boldsymbol{\Delta}^* = \frac{\beta}{K}(K_1, \delta, K_2, \delta, \ldots, K_n, \delta)^T \tag{B.19}$$

satisfies (B.15), (B.16) and (B.17) for all $i \in \{1, 2, \ldots, n\}$. Since $||\boldsymbol{\Delta}||_1 = 1$, we have that $\sum_{i=1}^{n} \Delta_i^* = \frac{\beta}{K}\sum_{i=1}^{n} K_i + \frac{\beta}{K} = 1$. Therefore, the vector defined in (B.19) with $\beta = (1 + \frac{n\delta}{K})^{-1}$ is the (unique) fixed point of the system.

B.5 Convergence of the Modified Proportional Fair Model

To see that the model in (3.12), intuitively, converges, let us make some observations. For the dynamics in (3.13) we showed that the fixed point is given by $\mathbf{w}_{\text{norm}}$. Therefore, when node $i-1$ fires, we have that $\Phi_{i-1} = 0$, $\Phi_i - \Psi_i = \beta K_i/K$, and

[3]Matrix $\mathbf{M}$ is primitive if $\exists k : \forall k' > k,\ \mathbf{M}^{k'} > 0$. Since matrices $\mathbf{M_i}$ have the 3-by-3 block shifted over the diagonal, the product $\mathbf{M} = \prod_{i=n}^{1} \mathbf{M}_i$ has positive elements on the upper-triangular part and on the last row and, thus, $\mathbf{M}^2 > 0$.

$\Psi_i - \Phi_{i-1} = \Psi_{i+1} - \Phi_i = \beta/(2K)$. Now, if we analyze the system in (3.12), we see that $\max(\Psi_i^{\text{target}}, \Psi_i/2) = \Psi_i^{\text{target}}$ and $\min(\Phi_i^{\text{target}}, (\Psi_{i+1} + \Phi_i)/2) = \Phi_i^{\text{target}}$ (where we omitted the time indices for brevity). In fact,

$$\begin{aligned} \Psi_i^{\text{target}} &= \beta \frac{1/2}{K_i + 1} \left(\frac{\beta}{K} + \beta \frac{K_i}{K} \right) = \frac{\beta}{2K} \frac{K_i + 1}{K_i + 1} = \frac{\beta}{2K} \\ &> \Psi_i/2 = \frac{\beta}{4K} \end{aligned}$$

and

$$\begin{aligned} \Phi_i^{\text{target}} &= \frac{K_i + 1/2}{K_i + 1} \left(\frac{\beta}{K} + \beta \frac{K_i}{K} \right) = \frac{\beta}{K} (K_i + 1/2) \\ &< (\Psi_{i+1} + \Phi_i)/2)/2 = \frac{\beta}{K}(K_i + 3/4). \end{aligned}$$

Therefore we have

$$\begin{aligned} \Psi(t_2^+) &= \alpha \Psi(t_2) + (1 - \alpha)\Psi(t_2) = \Psi(t_2) \\ \Phi(t_2^+) - \Psi(t_2^+) &= (1 - \alpha)\beta \frac{K_i}{K} + \alpha \left(\frac{\beta}{K}(K_i + 1/2) - \frac{\beta}{2K} \right) \\ &= \beta \frac{K_i}{K}. \end{aligned}$$

Thus, $\mathbf{w}_{\text{norm}}$ is also a fixed point of (3.12). Could there be other fixed points? The answer is no. In fact, consider the case where $\Phi_i - \Psi_i = \gamma \neq \beta K_i/K$, and $\Psi_i - \Phi_{i-1} = \Psi_{i+1} - \Phi_i = \gamma_0$. We could have $\max(\Psi_i^{\text{target}}, \Psi_i/2) \neq \Psi_i^{\text{target}}$ or $\min(\Phi_i^{\text{target}}, (\Psi_{i+1} + \Phi_i)/2) \neq \Phi_i^{\text{target}}$ or both.

Suppose, for instance, that $\max(\Psi_i^{\text{target}}, \Psi_i/2) = \Psi_i/2$ and $\min(\Phi_i^{\text{target}}, (\Psi_{i+1} + \Phi_i)/2) = (\Psi_{i+1} + \Phi_i)/2$. We would have

$$\begin{aligned} \Phi_i(t_2^+) - \Psi_i(t_2^+) &= (1 - \alpha)\gamma + \alpha \left(\frac{\Psi_{i+1} + \Phi_i}{2} - \frac{\Psi_i}{2} \right) \\ &= (1 - \alpha)\gamma + \alpha \frac{\Psi_{i+1} + \gamma}{2} = \gamma \end{aligned} \tag{B.20}$$

which is not true for any i and Ψ_{i+1}. Moreover, we have $\Psi_i(t_2^+) = \alpha \frac{\Psi_i(t_2)}{2} + (1 - \alpha)\Psi_i(t_2) = (1 - \alpha/2)\Psi_i(t_2) \neq \Psi_i(t_2)$. We can make similar observations for the

other two cases. In conclusion, model (3.12) has a unique fixed point, which is given by $\mathbf{w}_{\text{norm}}$. If we look at (3.12), we see that the dynamics the system follows, is equivalent to (3.13). The only difference is that, in this case, the clocks of node i are bounded by $\Psi_i/2$ and $(\Phi_i + \Psi_{i+1})/2$, but the behavior is the same, as well as the fixed point. Therefore, we provided an explanation on why model (3.12) converges, while preserving the interval gap between the two firings of each node as a collision free slot for the node transmission.

B.6 Proof of Lemma 1

The likelihood of each of different hypotheses can be evaluated by expressing the PMF with delta function:

$$\begin{aligned} P(Y = y|X = x) = {} & \delta[y - x](\delta[x - 1]\delta[y - 1]\beta + \delta[x]\delta[y]\alpha_{\text{b}}) \\ + {} & \delta[y - \bar{x}](\delta[x - 1]\delta[y - 1](1 - \beta_{\text{b},i}) \\ + {} & \delta[x]\delta[y](1 - \alpha_{\text{b}})) \end{aligned}$$

Now, because the symbols are binary, we can write the corresponding algebraic equations and after simplifications we get:

$$\begin{aligned} P(Y = y|X = x) = {} & g(x, y)(yx\beta_{\text{b},i} + \bar{y}\bar{x}\alpha_{\text{b}}) \\ & +\overline{g(x, y)}(\bar{y}x(1 - \beta_{\text{b},i}) + y\bar{x}(1 - \alpha_{\text{b}})). \end{aligned} \tag{B.21}$$

Where $g(x, y) = (yx + \bar{y}\bar{x})$, which is the complement of an ex-or function; The likelihood of a specific output given the input is the product of all these likelihoods

$$\begin{aligned} P(\{Y_n = y_n\}_{n=1}^N \ \mid \ & \{X_n = x_n\}_{n=1}^N) \\ & = \prod_{n=1}^{N} P(Y_n = y_n|X_n = x_n). \end{aligned}$$

Therefore, for the hypotheses H_s, H_e, H_0, the likelihood is computed as the previous expression specialized to the start and end beacons and to the all zero sequences respectively. If the input are random i.i.d data then the output remains a Bernoulli random variable with probability $p = \frac{\beta+1-\alpha_{\mathrm{b}}}{2}$. In this case the likelihood is given by

$$P(Y = y|H_d) = \delta[y-1]p + \delta[y](1-p) = yp + \bar{y}(1-p).$$

Thus, its log-likelihood becomes

$$\lambda(y_n|x_n) = y_n \log p + \bar{y}_n \log(1-p)$$

and for $p \neq 1$ we can write

$$\lambda(y_n|x_n) = y_n \log\left(\frac{p}{1-p}\right) + \log(1-p).$$

Computing the likelihood requires knowing the parameter $\beta_{\mathrm{b},i}$, which we said previously is transmitter dependent. Because $\beta_{\mathrm{b},i}$ is unknown it is reasonable to resort to a generalized likelihood ratio test (GLRT), that requires maximizing the likelihood of each hypothesis with respect to the unknown parameter $\beta_{\mathrm{b},i}$. We take the derivative of the log-likelihood with respect to $\beta_{\mathrm{b},i}$ and equate that to zero. After simple steps, the result for the estimate of $\beta_{\mathrm{b},i}$ is as given in the Lemma.

APPENDIX C

APPENDIX OF CHAPTER 4

C.1 Proof of Theorem 6

We need to prove that there always exists at least one sequence of n consecutive updates leading to an absorbing state. We pick the node who performed the k^{th} iteration and call it v. Consider the set of nodes defined by the following recursion

$$\mathcal{T}[k+l] = \mathcal{T}[k+l-1] \cup u \tag{C.1}$$

subject to the constraint

$$u \notin \mathcal{T}[k+l-1] : \exists\, e(u,v), v \in \mathcal{T}[k+l-1]$$

with $l = 0, 1, \ldots, n-1$ and $\mathcal{T}[k] = \{v\}$. Equation (C.1) states that, at each step, we choose a node that has at least one edge connected to at least one node in $\mathcal{T}[k+l-1]$. If there are, let us say, $1 \leq t < n$ such nodes, the probability that this situation actually happens is $t/n \geq 1/n > 0$. At iteration $k+1$, the probability that the new node u, added to $\mathcal{T}[k] = \{v\}$, chooses the same color as v is at least $1/Q_u = (\sum_m Q_{um})^{-1} > 0$, where Q_u is the total number of colors used by node u. Conditioned to this event, the probability that the second node, say z, added to $\mathcal{T}[k+1] = \{v, u\}$, chooses the same color is at least $1/Q_z = (\sum_m Q_{zm})^{-1} > 0$, and so on and so forth, for $n-1$ times. Therefore, given any initial color of v, the probability that after $n-1$ consecutive updates all nodes have that particular color is at least

$$\prod_{u \in \mathcal{N} - \{v\}} \frac{1}{nQ_u} > 0$$

so long as $n < \infty$. Since node v and iteration k are arbitrary, the color of node v could be any color in the set $\{0, 1, \ldots, C-1\}$, and, thus, the result follows.

C.2 Proof of Corollary 1

As described above, the state of the network is given by a $(C+1)n$-dimensional vector containing the n colors of the nodes, plus their *buffers*, i.e., the messages they received from the neighborhood from the last local update. Since this process is a Markov chain, and the absorbing set is a condition where all nodes have the same color and none of the other colors appear in their buffers, a sufficient condition for convergence is that there always exists a strictly positive probability of reaching the absorbing set [61]. From Theorem 6 we know that this situation always occurs, and, therefore, the result follows.

C.3 Proof of Corollary 2

In this case there is only one absorbing state, since the leader never changes its color. The proof of this result is equivalent to the proof of Theorem 2. Rather than looking at the system step by step, we must consider all indexes corresponding to the updates of the leader, i.e., $\mathcal{U}_1$. For each iteration $k \in \mathcal{U}_1$, Lemma 6 holds true, with $v = 1$ (the leader), and the result follows.

C.4 Proof of Theorem 7

Since the coalescing random walk process is the dual of the voter model, by Lemma 2,

$$E[T] = E[T_{crw}] \leq e \log(n+2) \max_{i,j} E_i T_j.$$

Moreover, denoting $\mathbf{M} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and by Lemma 3,

$$\begin{aligned}
\max_{i,j} E_i T_j &= \max_{i,j} 2|\mathcal{E}| \sum_{k=2}^{n} \frac{1}{1-\lambda_k(\mathbf{M})} \left(\frac{v_{kj}^2}{D_{jj}} - \frac{v_{kj}v_{ki}}{\sqrt{D_{jj}D_{ii}}} \right) \\
&\leq^{(a)} \frac{2|\mathcal{E}|}{1-\lambda_2(\mathbf{M})} \max_{i,j} \sum_{k=2}^{n} \left(\frac{v_{kj}^2}{D_{jj}} - \frac{v_{kj}v_{ki}}{\sqrt{D_{jj}D_{ii}}} \right) \\
&\leq^{(b)} \frac{2|\mathcal{E}|}{1-\lambda_2(\mathbf{M})} \left(\max_{j} \frac{1}{D_{jj}} + \max_{i,j} \sum_{k=2}^{n} \frac{-v_{kj}v_{ki}}{\sqrt{D_{jj}D_{ii}}} \right), \\
&=^{(c)} \frac{2|\mathcal{E}|}{1-\lambda_2(\mathbf{M})} \left(\max_{j} \frac{1}{D_{jj}} + \max_{i,j} \frac{v_{1j}v_{1i}}{\sqrt{D_{jj}D_{ii}}} \right), \\
&\leq^{(d)} \frac{4|\mathcal{E}|}{1-\lambda_2(\mathbf{M})} \max_{j} \frac{1}{D_{jj}},
\end{aligned}$$

where (a) follows from the fact that $\lambda_2(\mathbf{M})$ is the second largest eigenvalue, (b) is due to $\sum_{k=1}^{n} v_{kj}^2 = 1$, (c) follows from $\sum_{k=1}^{n} v_{kj}v_{ki} = 0$ and $v_{1j} \geq 0$, and (d) from $v_{1j}v_{1i} \leq 1$. If we combine the upper bound with Lemma 2, we obtain the final bound:

$$E[T] \leq \frac{4e\log(n+2)|\mathcal{E}|}{1-\lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})} \max_{j} D_{jj}^{-1}$$

Finally, we note that $\lambda_k(.)$'s and v_k's are well defined since $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is a real symmetric matrix.

C.5 Proof of Theorem 8

We first note that for $r \geq (4\log(n)/n)^{1/2}$ and large enough n, the graph will be connected with high probability (w.h.p.). Moreover, there exists $n' \geq 0$ such that for all $n \in [n', \infty)$ the degree of every node in the network is equal to some $\alpha(n)$ w.h.p. [14]. Thus, the ratio of $|\mathcal{E}|/\min_{jj} D_{jj}$ is equal to n w.h.p. Moreover, the term $[1-\lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})]^{-1}$ upper bounded by is $1/r^2$ for large enough N [14]. Combining these results with Theorem 8, the result follows.

C.6 Proof of Theorem 9

Using (4.8), one can show that

$$P\left(d_i \leq np + \delta np\right) \geq 1 - e^{\frac{np\delta^2}{3}}.$$

Moreover, if we choose $\delta = 3\log(n^2)/np$:

$$P\left(d_i \leq Np + \sqrt{3np\log(n)}\right) \geq 1 - \frac{1}{n^2}.$$

Since above equation holds for all $i \in \mathcal{V}$, then:

$$P\left(|\mathcal{E}| \leq n^2 p + n\sqrt{3np\log(n)}\right) \geq 1 - \frac{1}{n^2}. \tag{C.2}$$

To bound $\max_j D_{jj}^{-1}$, we first note that $D_{jj} = |\backslash_j| = d_j$. Then, by (4.9):

$$P\left(d_j \geq np + \delta np\right) \geq 1 - \frac{1}{n^2}.$$

By choosing $\delta = 2\log(n^2)/np$:

$$P\left(\min_j D_{jj}^{-1} \leq \left(np - \sqrt{2np\log(n)}\right)^{-1}\right) \geq 1 - \frac{1}{n^2} \tag{C.3}$$

To bound $1 - \lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})$, we utilize Theorem 3.6 in [18]. By noting that $w_{min} = \bar{w} = np$ in our case and assuming $np >> \log^2(n)$, the bound simplifies to:

$$|1 - \lambda_{n-1}(I - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})| \leq \quad (1 + o(1))\frac{4}{np} + g(n)\frac{\log^2(n)}{np},$$

where $g(n)$ is an a function tending to infinity arbitrary slowly. Since $\lambda_2 = 1 - \lambda_{n-1}(I - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})$, we can bound the last term in (4.6) as:

$$\frac{1}{1 - \lambda_2(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})} \leq \left(1 - \frac{8}{np} - \frac{g(n)\log^2(n)}{np}\right)^{-1}. \tag{C.4}$$

Combining (C.2), (C.3) and (C.4), our result follows.

BIBLIOGRAPHY

[1] Bluetooth specification. In *SIG Bluetooth - http://www. bluetooth. com.*

[2] Zigbee specification. In *ZB Alliance - ZigBee Document 053474r06*, 2004.

[3] J.A. Acebrón, L.L. Bonilla, C. J. Pérez Vicente, F. Ritort, and R. Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Rev. Mod. Phys.*, 77(1), April 2005.

[4] D.J. Aldous. Random walks on finite groups and rapidly mixing markov chains. In *Seminaire de Probabilities XVII*. Springer, 1983.

[5] A. Apsel, R. Dokania, and X. Wang. Ultra-low power radios for ad-hoc networks. In *IEEE International Symposium on Circuits and Systems*, 2009.

[6] T.C. Aysal, M.J. Coates, and M.G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56(10):4905–4918, October 2008.

[7] T.C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*. to appear.

[8] T.C. Aysal, M. E. Yildiz, and A. Scaglione. Broadcast gossip algorithms. In *Technical Report*, 2007.

[9] S. Barbarossa and G. Scutari. Bio-inspired sensor network design: Distributed decision through self-synchronization. *IEEE Signal Processing Magazine*, 2007.

[10] M. Bennett, M. Schatz, H. Rockwood, and K. Wiesenfeld. Huygens clocks. *Proc. R. Soc. Lond. A.*, 2002.

[11] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[12] V.D. Blondel, J.M. Hendrickx, A. Olshevsky, and J.N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proc. of IEEE Conference on Decision and Control*, pages 2996–3000, December 2005.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proc. INFOCOM*, 2005.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *INFOCOM*, 2005.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algotirhms. In *IEEE/ACM Transactions on Networking*, 2006.

[16] J. Buck. Synchronous rhytmic flashing of fireflies. In *Quart. Rev Biol.*, 1988.

[17] J. R. Carpenter. Decentralized control of satellite formations. *Intl. J. of Robust and Nonlinear Control*, 12:141–161, 2002.

[18] Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America*, 100, May 2003.

[19] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal. Desync: Self-organizing desynchronization and tdma on wireless sensor networks. In *International Conference on Information Processing in Sensor Networks (IPSN)*, April 2007.

[20] M. Dorigo. *Ant Colony Optimization*. The MIT Press, 2004.

[21] R.C. Eberhart, Y. Shi, and J. Kennedy. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[22] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *SIGOPS Oper. Syst. Rev.*, 2002.

[23] L. Fang, P.J. Antsaklis, and A. Tzimas. Asynchronous consensus protocols: preliminary results, simulations and open questions. In *44th IEEE Conference on Decision and Control*, pages 2194–2199, Dec. 2005.

[24] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Trans. on Automatic Control*, 49:1465–1476, 2004.

[25] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1996.

[26] S. Gregory. Finding overlapping communities in networks by label propagation. *arXiv:0910.5516v1*, (Submitted) 2009.

[27] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2), 2000.

[28] F.E. Hanson. Cooperative studies of firefly pacemakers. In *Fed. Proc*, 1978.

[29] Y. W. Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applications. *IEEE Journal of Selected Areas in Communications, Special Issue on Advances in Military Wireless Communications*, 23(5):1085–1099, 2005.

[30] Y.-W. P. Hong and A. Scaglione. Time synchronization and reach-back communications with pulse-coupled oscillators for uwb wireless ad hoc networks. In *IEEE Conference on Ultra Wideband Systems and Technologies (UWBST 2003)*, 2003.

[31] F.C. Hoppensteadt and E.M. Izhikevich. Weakly connected neural networks. In *Springer-Verlag*, 1998.

[32] Crossbow http://www.tinyos.net/.

[33] TinyOS http://www.tinyos.net/.

[34] A. Hu and S.D. Servetto. On the scalability of cooperative time synchronization in pulse-connected networks. In *IEEE Transaction on Information Theory*, 2006.

[35] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. on Automatic Control*, 48:988–1001, 2003.

[36] L. Jiang and J. Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. Technical report, University of California at Berkeley, 2008.

[37] Emil Jovanov. A survey of power efficient technologies for wireless body area networks. In *Proc. of the 30th Annual Internation Conference of the IEEE Engineering in Medicine and Biology Society*, Vancouver, Canada, 2008.

[38] J.W. Jung, A. Kailas, M.A. Ingram, and E.M. Popovici. An evaluation of cooperation transmission considering practical energy models and passive reception. In *ISABEL 2008*.

[39] Akshay Kashyap, Tamer Başar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

[40] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2003.

[41] S. Kirti, A. Scaglione, and R. J. Thomas. A scalable wireless communication architecture for average consensus. In *Proc. IEEE Conference on Decision and Control*, pages 32 – 37, Dec. 2007.

[42] D. J. Klein, P. Lee, K. A. Morgansen, and T. Javidi. Integration of communication and control using discrete time kuramoto models for multivehicle coordination over broadcast networks. *IEEE JSAC*, 26(4):695–705, 2008.

[43] D. J. Klein and K. A. Morgansen. Controlled collective motion for trajectory tracking. In *Proc. of the IEEE American Control Conference*, June 2006.

[44] D.J. Klein, P.K. Bettale, B.I. Triplett, and K.A. Morgansen. Autonomous underwater multivehicle control with limited communication: Theory and experiment. In *Proc. of the Second IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*, 2008.

[45] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis. Collective motion, sensor networks and ocean sampling. *Proceedings of the IEEE*, 95(1):48–74, 2007.

[46] Q. Li and D. Rus. Global clock synchronization in sensor networks. In *INFOCOM*, 2004.

[47] T.M. Liggett. *Interacting Particle Systems*. Srpinger-Verlag, 1985.

[48] X. Liu and T. Murata. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *arXiv:0910.1154v2*, 2009.

[49] L. Lovasz. Random walks on graphs: A survey. *Combinatorics*, 1993.

[50] D. Lucarelli and I.-J. Wang. Decentralized synchronization protocols with nearest neighbor communication. In *Sensys*, 2004.

[51] Xavier Lurton. *Underwater Acoustics: An Introduction*. Springer, 2003.

[52] E. Mallada and K. Tang. Synchronization of coupled oscillators. In *ITA*, 2010.

[53] R. Mangharam, A. Rowe, and R. Rajkumar. Firefly: A cross-layer platform for real-time sensor networks. In *Real Time Systems Journal*, Camera ready submission in progress.

[54] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[55] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray. Asynchronous distributed averaging on communication networks. In *IEEE Transaction on Networking*, 2007.

[56] R.E. Mirollo and S.H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math.*, 50(6):1645–1662, 1990.

[57] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *In The 19th IEEE Symposium on Reliable Distributed Systems, SRDS'00*, 2000.

[58] N. Moshtagh, N. Michael, A. Jadbabaie, and K. Daniilidis. Vision-based, distributed control laws for motion coordination of nonholonomic robots. *IEEE Journal on Robotics*. to appear.

[59] L. Murray. Stability of multiagent systems with time-dependent communication links. In *IEEE trans. on Automatic Control*, 2005.

[60] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E 74, 036104*, 2006.

[61] J.R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[62] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. In *IEEE Trans. on Automatic Control*, 2006.

[63] R. Olfati-Saber and M. Murray. Consensus problems in networks of agents with switching topologies and time-delays. In *IEEE Trans. on Automatic Controls*, 2004.

[64] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents

with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49(9):1520 – 1533, Sept. 2004.

[65] A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. On distributed averaging algorithms and quantization effects. Technical report, MIT, 2007.

[66] S. Paquelet, L.M. Aubert, and B. Uguen. An impulse radio asynchronous transceiver for high data rates. In *Ultra Wideband Systems, 2004. Joint with Conference on Ultrawideband Systems and Technologies. Joint UWBST and IWUWBS.*, 2004.

[67] J. Partan, J. Kurose, and B. Levine. A survey of practical issues in underwater networks. In *Proceedings of the 1st ACM international workshop on Underwater networks*, pages 17 – 24, Los Angeles, CA, 2006.

[68] Ankit Patel, Julius Degesys, and Radhika Nagpal. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. In *IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, July 2007.

[69] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *NSDI*, 2007.

[70] C. Peskin. Mathematical aspects of heart physiology. *Institute of Mathematical Sciences*, 1975.

[71] C.S. Peskin. Mathematical aspects of heart physiology. In *Courant Institute of Mathematical Sciences*, 1975.

[72] U.N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *arXiv:0709.2938v1 [physics.soc-ph]*, 2007.

[73] K. Romer. Time synchronization in ad hoc networks. In *MobiHoc*, 2001.

[74] H. Sawada, T. Aoyagi, J-I Takada, K. Y. Yazdandoost, and R. Kohno. Channel models between body surface and wireless access point for UWB band. In *IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)*, 2008.

[75] A. Scaglione, D. Goeckel, and J. N. Laneman. Cooperative communications in mobile ad-hoc networks: Rethinking the link abstraction. In *IEEE Signal*

Processing Magazine: Special Issue on Signal Processing for Wireless Ad Hoc Communication Networks, 2006.

[76] A. Scaglione and Y.W. Hong. Opportunistic large arrays: Cooperative transmission in wireless multihop ad hoc networks to reach far distances. In *IEEE Trans. on Signal Processing*, 2003.

[77] L. Schwiebert, L..S. Gupta, and Jennifer Weinmann. Research challenges in wireless networks of biomedical sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001.

[78] M. L. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC'03*, 2003.

[79] M. Stojanovic. Underwater acoustic communications: Design considerations on the physical layer. In *Proc. IEEE / IFIP Fifth Annual Conference on Wireless On demand Network Systems and Services*, 2008.

[80] W. Su and I.F. Akyildiz. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans. on Networking*, 2005.

[81] G. Tibely and J. Kertesz. On the equivalence of the label propagation method of community detection and a potts model approach. *Physica A*, 2008.

[82] A. Tyrrell, G. Auer, and C. Bettstetter. Biologically inspired synchronization for wireless networks. *Series: Studies in Computational Intelligence, Springer*, 2007.

[83] A. Tyrrell, G. Auer, and C. Bettstetter. Biologically inspired synchronization for wireless networks. In *Advances in biologically inspired information systems*, 2007.

[84] X. Wang and A. Apsel. Pulse coupled oscillator synchronization for communications in uwb wireless transceivers. In *MWSCAS*, 2007.

[85] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05*, 2005.

[86] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *Systems and Control Letters*, 2004.

[87] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proc. International Conference on Information Processing in Sensor Networks*, Los Angeles, CA, April 2005.

[88] M. E. Yildiz and A. Scaglione. Coding with side information for rate constrained consensus. *IEEE Trans. on Signal Processing*, 56(8):3753 – 3764, August 2008.